SOLAR ENERGY HARVESTING AND SOFTWARE ENHANCEMENTS FOR AUTONOMOUS WIRELESS SMART SENSOR NETWORKS

BY

TIMOTHY ISAAC MILLER

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Civil Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2009

Urbana, Illinois

Adviser:

Professor Bill F. Spencer, Jr.

ABSTRACT

Civil infrastructure is the backbone of modern society, and maintaining said infrastructure is critical in maintaining healthy society. Wireless smart sensors (WSSs) provide a means to effectively monitor the performance of buildings and bridges to improve maintenance practices, minimize the costs of repair, and improve public safety through a process called structural health monitoring (SHM). WSSs, traditionally powered by batteries, are limited in the length of time they can operate autonomously. The frequent need to change batteries in the networks can drive up maintenance costs and diminish the advantage first realized with WSSs. Efforts have been made to minimize the power consumption of WSSs operating in SHM networks, but there have been a limited number of new power supply options, such as energy harvesting, used in full-scale SHM applications. This research develops a solar energy harvesting system to provide power to Imote2 WSS platform and increase the long-term autonomy of wireless smart sensor networks (WSSNs). The approach is validated on a cable stayed bridge in South Korea. Additionally, software enhancements are introduced to allow sensor data to be stored in non-volatile memory, potentially further enhancing the efficacy of WSSNs. This research has resulted in greater overall autonomy of WSSNs.

Contents

Chapter 1 Introduction

Civil infrastructure is the backbone of modern society, and maintaining the safety and performance of that infrastructure is critical to maintaining a healthy society. Recent catastrophic structural failures (I-35W bridge in Minneapolis, 2007), however, have exposed the shortcomings of the maintenance efforts that have been in place for some years. These recent failures have increased the public's overall demand for effective monitoring of civil infrastructure. Effectively monitoring the performance of buildings and bridges provides the means to improve maintenance practices, minimize the costs of repair, and improve public safety. Structural health monitoring (SHM) involves the measurement of structural response and using that response data to determine the structural condition, making SHM useful in numerous ways. For instance, the information gained from SHM may be used to update idealized structural models, increasing the accuracy of predicted responses to extreme loading events like earthquakes and tsunamis. Additionally, SHM systems can be used as real-time monitoring systems to measure structural behavior before, during, and after a disastrous event. The information collected over this time period can be used in damage detection algorithms to determine the post-event structural integrity.

In recent years, efforts have been focused on this technique of using collected data to determine the state of a structure. To determine the state of the structure, the algorithms combine the measured structural response with information about the input excitation and the structural model, and try to determine if there have been any distinguishable changes in the structural condition (i.e. if the structure has been damaged). This type of analysis is effective for long term monitoring to detect changes in condition over time, and it is also effective to assess the structural condition after a high-impact event like an earthquake or tsunami. Either way, the result is an expedited detection of damage, thus allowing for the execution of evacuation procedures and repair processes, ultimately improving public safety and reducing the cost of structural upkeep.

High-fidelity sensor data is required to build the structural models necessary to make accurate determinations of a structure's condition. Moreover, many potential structural problems, such as buckling and fracture, occur relatively locally. With the intrinsically local

nature of damage, compounded by the need for high-fidelity data, a dense array of sensors is needed to capture any changes in structural response resulting from damage. Installing such a dense array of sensors on a large structure is incredibly expensive and difficult if the sensors are the wired type traditionally used in structural health monitoring systems. These wired systems have sensors distributed around the structure with wires to provide power and wires to transmit data, resulting in a cumbersome arrangement. Often, the expense and difficulty of installing traditional wired systems make them impractical options.

To exemplify the point regarding the high cost of wired SHM systems, Çelebi (2002) has estimated the cost per sensor channel (with the sensor, data acquisition system and installation) to be approximately $4,000 for systems with 12 to 18 channels. Additionally, The Bill Emerson Memorial Bridge in Missouri has 84 accelerometer channels with an average cost per channel of over $15,000, including installation (Çelebi et al. 2004). With such high cost figures, it is easy to see that wired SHM systems are not practical in many projects, especially those with limited budgets or limited time to spend on installation.

With the current state-of-the-art in wireless technology and embedded processing, using wireless smart sensor networks (WSSNs) for SHM applications is a feasible, and often preferred, alternative to traditional wired systems. By eliminating the wires for power and data transmission, the installation time of the WSSN is significantly lower than that of the wired systems. With reduced installation times, the cost of implementation is also reduced.

Wireless smart sensors even utilize onboard computational capacity to allow data processing within the network, rather than all at a central location. By processing data within the network, communication time is shortened, but valuable information about the structural condition is still communicated to the user.

Despite the numerous advantages of WSSNs, there are myriad challenges when the networks are implemented for monitoring purposes. In an ideal setting, the WSSN would operate autonomously, with minimal input from the human user. With autonomous operation, the WSSN can be utilized by users with minimal training in the operation and maintenance of such networks. To properly achieve autonomous operation, however, it is important that the network be functioning for extended periods of time with virtually no maintenance

requirements. Since one of the primary methods of powering WSSNs is through the use of batteries, it is critical to implement a smart power management strategy to avoid power failure. An effective strategy will address power management from the perspective of both power supply and power consumption. Without smart power management strategies, the frequent replacement of batteries will cause exorbitant maintenance. The costs incurred from frequently swapping batteries can offset the cost advantage first realized with the installation of a WSSN.

Even with the implementation of critical power management strategies in WSSNs, it is also beneficial to introduce redundancy into the systems to account for unforeseen failures. For instance, to increase reliability, the network should be able to operate without the presence of a permanent base station sensor or data acquisition system. By storing sensor data on remote nodes, rather than requiring data to be sent to a data acquisition system, the network is allowed to continue monitoring in the event of unanticipated failures, such as communication loss or power failure.

Efforts have been made to limit the amount of power consumed in high-performing wireless smart sensors. In Rice and Spencer (2009), a unique service that puts the sensor node into sleep mode is utilized to reduce power consumption. Even with this sleep mode, however, more efforts need to be made on the energy supply side of power management. In a test conducted at the University of Illinois utilizing the sleep mode, the network remained operational for 57 days. The only limitation to extending that timeframe was the battery life of the sensor nodes (Rice and Spencer 2009).

The objective of this research is to enhance the efficacy of the autonomous operation of WSSNs in SHM systems. This research focuses mainly on two aspects that concern the autonomy of WSSNs: 1) The development of a solar energy harvesting system to extend the lifetime of WSSNs, and 2) the development of a software application to store sensor data in non-volatile, flash memory. The following paragraphs provide an outline for the contents of this report.

Chapter 2 provides a comparison between various smart sensor platforms, followed by a discussion of previous efforts to minimize the power consumption of these platforms. An

introduction to energy harvesting is given. Finally, an overview of the traditional sensor data collection scheme is provided, along with its limitations.

Chapter 3 gives an overview of solar energy harvesting. The basic functions and characteristics of solar panels are given. Various rechargeable battery chemistry types are discussed. The advantages of each chemistry type are given.

Chapter 4 discusses the process of implementing a solar energy harvesting system on the Imote2. Selection of a solar panel and a rechargeable battery is made. The various hardware modifications necessary for implementation are presented. The software to control battery charging is discussed.

Chapter 5 introduces an application that stores sensor data to flash, named *RemoteFlashSensing*. Since most SHM applications require sensing, an application called *SensingUnit* is also discussed. The way in which *RemoteFlashSensing* operates, as well as how to operate it, is discussed in this chapter.

Chapter 6 validates the solar energy harvesting system developed for this research on the Jindo Bridge in South Korea. The test demonstrates the viability of solar energy harvesting for WSSNs.

Chapter 7 gives a summary of the research presented in this report, and discusses future studies that may be pursued to increase the effectiveness of autonomous WSSNs.

Chapter 2 Background

This chapter focuses on the background information and the motivation for this research. The beginning of the chapter provides a comparison between multiple wireless sensor platforms that can be used in structural health monitoring (SHM) systems, including their processor speed, memory size, power consumption, and power source. Following this comparison, previous efforts in power consumption minimization are discussed. Next, an introduction to energy harvesting is given, and a survey of previous research regarding energy harvesting for wireless sensor networks is provided. The background on energy harvesting explores the performance differences between various harvesting technologies. Then, an overview of the current sensor data storage method is given, and some shortcomings of this method are presented. The goal of this chapter is to identify the remaining challenges to achieving autonomous operation of wireless smart sensor networks.

2.1 Wireless smart sensor platform comparison

Rice and Spencer (2009) provide a detailed comparison of the current wireless smart sensor (WSS) platforms, which includes references to papers summarizing the development of smart sensor technology, as well as wireless platforms that have been proposed for structural health monitoring (SHM) (e.g., Spencer et al. 2004; Lynch and Loh 2006). The wireless platforms are divided into two basic categories: academic prototypes and commercially available platforms.

The academic prototypes vary greatly in terms of processor speeds, memory capacities, operating frequencies, and power consumption. Most of them utilize an onboard analog-to-digital converter (ADC), which can limit the sensor design options for these platforms. The largest disadvantage of the academic prototypes, however, is that they use proprietary technology without allowing access by a larger community. This proprietary nature limits the contributions that can be made by outsiders, and ultimately impedes the widespread use of these academic prototypes.

In contrast, most commercially available wireless sensor platforms allow access by a larger community, though their hardware design and operating software are not exposed to the user. The openness of the commercially available platforms encourages contributions from outsiders, and ultimately makes them more likely to be applied on a large scale. This report will

provide a summary of the characteristics of four commercially available wireless platforms. A more detailed overview can be found in Rice and Spencer (2009).

One of the first commercially available sensor platforms with open source hardware and software was the Rene Mote in 1999, which eventually led to the third generation of Mote: the Mica. Mica Mote, as it was named, has an 8-bit microcontroller with a 4 MHz processor (CPU), which possesses 128 kB of ROM (read-only memory) and 4 kB of RAM (random access memory). 512 kB of non-volatile (flash) memory is also included. The Mica Mote suffers from poor radio communication performance, however, and an upgraded model was necessary.

In 2002, Crossbow introduced the Mica2, which improved upon the radio performance of the previous Mica Mote (Crossbow 2007b). Additional upgrades to the microcontroller were included, as well. The MicaZ – the most recent version of the Mica Mote from Crossbow – is similar to the Mica2, but it features an 802.15.4 2.4 GHz radio and the module's physical dimensions are smaller.

Desiring a smart sensor module with lower power consumption than that of the Mica modules, researchers at Berkeley developed the Telos mote in 2004 (Polastre et al. 2005). The researchers chose a microcontroller with low power requirements and quick transition times between operating modes. The Telos, much like the Mica motes, features an onboard ADC, a potential limitation in sensor design flexibility.

Intel released a new smart sensor platform in 2003 called the iMote, which was the culmination of collaboration between Intel Research Berkeley Laboratory and UC Berkeley (Kling 2003). The second generation of the Intel Mote, called the Imote2, was introduced in more recent years (Kling et al. 2005). The Imote2 functions on a low-power X-scale processor from Intel (PXA27x). The processor speed can be adjusted based on application needs, allowing peak performance without consuming unnecessary amounts of power. The Imote2 also lacks an onboard ADC, leaving significant flexibility in the sensor design for this platform.

A summary of these commercially available, open-source wireless sensor platforms is presented in Table 2.1. As can be seen, the Imote2 offers superior performance in processor speed to it contemporaries, and offers more RAM. The Imote2's superior features make it the

most ideal commercially available, open-source WSS platform that is feasible for the high computational demands of many SHM applications (Rice and Spencer 2009).

Table 2.1 Comparison of commercially available smart sensor platforms.

	Mica2 (Crossbow)	MicaZ (Crossbow)	Telos(B)/Tmote Sky (MoteIV[1])	Imote2 (Crossbow)
Processor	ATmega128L	ATmega128L	TIMSP430	XScalePXA271
Bus Size (bits)	8	8	16	32
Processor Speed (MHz)	7.373	7.373	8	13 – 416
Program Flash (bytes)	128 K	128 K	48 K	32 M
EEPROM (bytes)	512 K	512 K	N/A	N/A
RAM (bytes)	4 K	4 K	1024 K	256 K SRAM 32 M SDRAM
Radio Chip	CC1000	CC2420	CC2420	CC2420
ADC resolution (bits)	10	10	12	N/A
ADC channels	8	8	8	N/A
Digital Interface	DIO, I2C, SPI	DIO, I2C, SPI	I2C, SPI, UART, USART	I2C, SPI, GPIO, UART, PWM, SDIO, USB
Active Power (mW)	24	24	10	44 @ 13 MHz 116 @ 104 MHz 570 @ 416 MHz
Sleep Power (µW)	75	75	8	100
Primary Battery	2 x AA	2 x AA	2 x AA	3 x AAA

1. Now Sentilla

2.2 Power consumption and management of the Imote2

The Imote2 is the only commercially available smart sensor platform that meets the needs of high data throughput applications such as structural health monitoring (SHM); however, these performance characteristics are not achieved without sacrificing in another area: power consumption. As seen in Table 2.1 in the previous section, the Imote2 draws more power than its contemporaries when in an active state. The increased power consumption hints at the necessity for smart power management schemes. Without an intelligent approach to power management, the traditional form of power supply on the Imote2 – non-rechargeable batteries – will limit the feasibility of long term SHM applications. These limitations stem from the necessity to swap batteries on a frequent basis, perhaps on modules located in hard-to-access areas. To be effective, the power management schemes should focus on minimizing power consumption as well as perpetuating the supply of power.

Rice and Spencer (2009) detail a collaborative project between researchers in civil engineering and computer science at the University of Illinois at Urbana-Champaign, called the Illinois Structural Health Monitoring Project (ISHMP). The objective of the ISHMP is to eliminate the complexity associated with developing wireless smart sensor network (WSSN) applications. The ISHMP developed a framework for SHM which provides a number of services necessary to provide high-quality sensor data and communicate it effectively throughout the network. It also provides a broad selection of numerical algorithms. This software is open-source and is available at http://shm.cs.uiuc.edu/.

For the applications available from the ISHMP, there are five primary power states of the Imote2 with a sensor board attached (Rice and Spencer 2009):

1. Deep sleep mode
2. Startup – initial state when Imote2 is turned on or wakes from deep sleep mode
3. Imote2 processor @ 13 MHz (lowest operating speed)
4. Imote2 processor @ 104 MHz (intermediate operating speed)
5. Sensing with the Imote2 processor @ 104 MHz

The current draw – and thus the power draw – in each of the power states is dependent upon the hardware used. In 2009, researchers at the University of Illinois working on the ISHMP

introduced a high-fidelity sensor board for use with the Imote2, called the SHM-A board (Rice and Spencer 2009). This new sensor board addressed many of the shortcomings of the board provided by Crossbow called the ITS400CA/B. The SHM-A sensor board draws approximately twice the amount of current as the ITS400CA/B while sensing, but the power is shut off to most portions of the SHM-A board when it is not sensing. The ITS400CA is powered the entire time that the Imote2 is not in deep sleep mode. Consideration needs to be given to the sensor board in use when considering a power management strategy.

The other main hardware component to consider is the type of battery board being used with the Imote2. In Rice and Spencer (2009), the main two battery boards under consideration were the first generation Intel battery board and the newer Crossbow battery board (IBB2400CA). The primary difference between the two boards is that the Intel battery board features a buck-boost regulator, which regulates the power to the Imote2 at ~3.2 V. The Crossbow board lacks any such regulator. As a result of the regulator, the Intel battery board allows continued operation of the Imote2 even when batteries are nearly depleted, but it consumes power at all times (even when the Imote2 is in sleep mode). The Crossbow battery board does not consume power when the Imote2 is sleeping, but it cannot continue operating once the battery voltage drops below 3.2V. Both battery boards were studied in Rice and Spencer (2009) to demonstrate the effects yielded from different hardware selections. Figure 2.1 shows the current draw in each of the 5 power states.

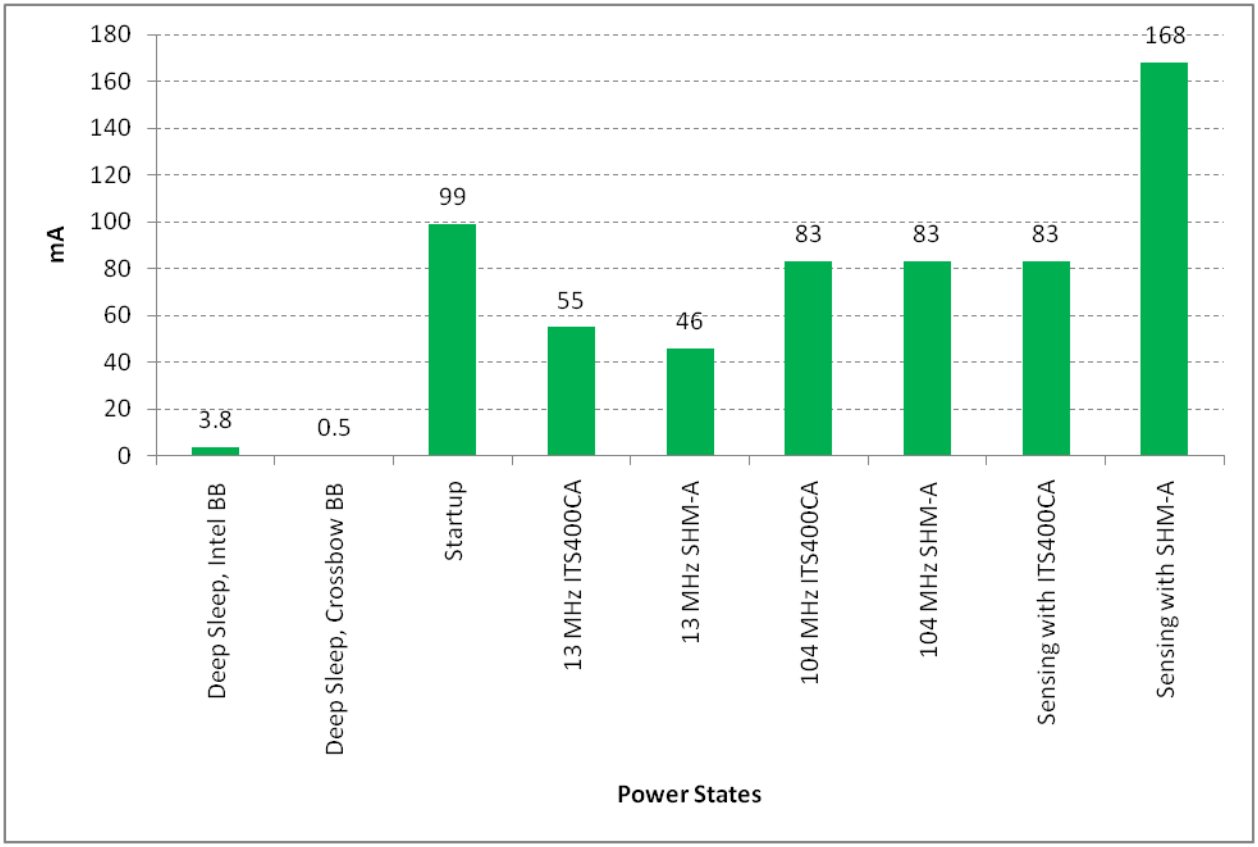


Figure 2.1 Approximate current consumption of the power states for the Imote2 with different battery and sensor boards (Rice and Spencer 2009).

The current consumption values presented in Figure 2.1 can be used to determine the impact of various application parameters on the total power consumption of each node. Furthermore, a number of software/application parameters can be varied to minimize power consumption. The parameters, some of which are subject to change by the user, are presented here in Table 2.2 as a matter of completeness.

Table 2.2 Power management optimization parameters (Rice and Spencer 2009).

Category	Parameter	Description
Network parameters	Network size	Number of nodes in network
	Sentry network size	Number of nodes involved in *ThresholdSentry*
Sensing parameters	Sampling Rate	*RemoteSensing* sampling rate
	Number of points	Number of data points in *RemoteSensing*
	Number of channels	Number of channels measured in *RemoteSensing*
	Number of *RemoteSensing* events	Number of *RemoteSensing* events per day
RemoteSensing wait times	Synchronization wait time	Time before sensing starts to synchronize the network
	Extra wait time	Extra time added to the total time the base station node waits between sending the sensing command and requesting data
	Extra sensing delay per node	Additional extra wait time per node to account for longer communication times in larger network
SnoozeAlarm times	Sleep interval	Sleep interval in *SnoozeAlarm* mode
	Listen interval	Short wake/listen time in *SnoozeAlarm* mode
ThresholdSentry times	Check interval	Time between sentry node checks
	Check sensing time	Time sentry node senses when checking data

As evidenced in Table 2.2, two of the categories to optimize for power consumption are the sleep interval and listen interval for *SnoozeAlarm*, which is a mode developed to save power by putting nodes in deep sleep mode when they are not involved in another application (Rice and Spencer 2009). In *SnoozeAlarm*, the node sleeps for a period of time and then wakes for a short listening period. This sleeping/waking process is repeated until a command is received to participate in another function. Figure 2.2 depicts the operation of a node using

SnoozeAlarm mode. By utilizing *SnoozeAlarm* on the Imote2, the power consumption is dramatically decreased as compared to that of the Imote2 without *SnoozeAlarm* enabled.

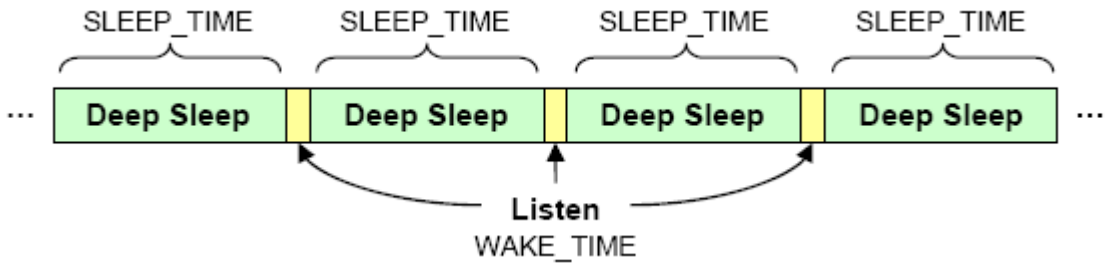


Figure 2.2 *SnoozeAlarm* timing (Rice and Spencer 2009).

Beyond using *SnoozeAlarm* to limit the power consumption of the Imote2, Rice and Spencer (2009) also take advantage of the fact that the Imote2 features a scalable processor. By decreasing the speed of the processor during less demanding tasks – such as time synchronization – and then increasing the speed only when necessary – such as during sensing – the power consumption is minimized.

After implementing the previously mentioned power-saving measures, Rice and Spencer (2009) performed a study to estimate the battery life of an Imote2 running in a WSSN. In order to remain consistent with full-scale validation test, the authors estimated the life of 3 D-cell alkaline batteries. For these batteries, it was experimentally determined that the voltage drops significantly during sensing – when the current draw is large – and then recovers almost completely to the original voltage level after sensing is completed. This battery behavior is important to consider because the lowest voltage experience during sensing must remain above the minimum voltage required by the sensor board/battery board combination being used on the Imote2. Figure 2.3 depicts the voltage drop behavior during sensing.

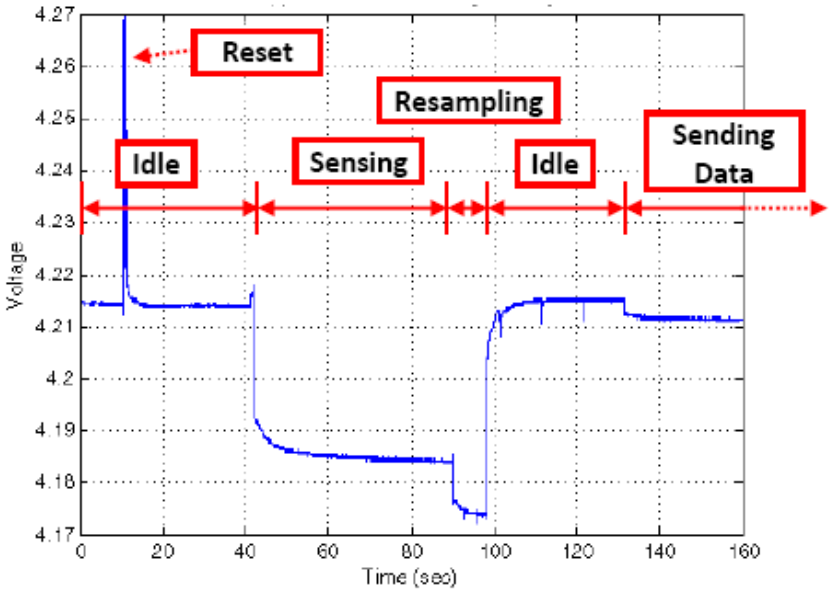


Figure 2.3 *RemoteSensing* voltage drop and recovery (Rice and Spencer 2009).

The values in Table 2.3 were determined experimentally, and show the maximum and minimum battery voltages for all combinations of battery boards and sensor boards.

Table 2.3 Voltage ranges for each battery board/sensor board combination (Rice and Spencer 2009).

Battery Board	V_{min_ITS}	V_{max}	V_{min_SHM-A}	V_{range_SHM-A}	V_{range_ITS}
Intel	2.9	4.8	2.9	1.9	1.9
Crossbow	3.6	4.7	3.8	0.9	1.3

The parameters used in the battery life study are shown in Table 2.4, and the estimated battery life of 3 D-cell alkaline batteries is shown in Table 2.5.

Table 2.4 Network and sensing parameters for power consumption parameter evaluation (Rice and Spencer 2009).

Parameter	Value
Network size, *n*	30
Number of *RemoteSensing* events per day, n_{RS}	2
Sampling rate, *fs*	100 Hz
Channels sampled, *ch*	3
Number of data points, *d*	10,000
Node order number, *i*	[0.5n]
Number of sentry nodes, n_s	[0.25n]
ThresholdSentry check interval, t_{TS_int}	15 min
ThresholdSentry check time, t_{TS_check}	20 sec
SnoozeAlarm duty cycle, δ_{SA}	4.76%
Startup duty cycle, δ_{start}	6.7%

Table 2.5 Estimated service life (days) for 3 D-cell batteries with various hardware combinations (Rice and Spencer 2009).

	Sensor Board			
Battery Board	SHM-A		ITS400CA	
	Sentry	Non-Sentry	Sentry	Non-Sentry
Intel	63	66	62	63
Crossbow	49	52	57	59

While the previous efforts to minimize power consumption have been significant, they are only part of the solution to achieving autonomous WSSNs. As evidenced in Table 2.5, even with power saving measures in place, the network can run uninterrupted for roughly 60 days. Requiring a change of batteries every two months is not ideal for an autonomous network. The key, therefore, is to couple the power saving measures with innovative power supply measures.

An effective method to harvest ambient energy and supply it to the nodes in a WSSN is necessary to achieve full autonomous operation.

2.3 Energy Harvesting

Though minimizing power consumption can significantly extend the battery life of a wireless smart sensor (WSS), ambient energy must also be harnessed and utilized by the WSS to ultimately create a perpetual power supply. This process of extracting ambient energy and converting it into usable energy is known as energy harvesting. Implementing effective energy harvesting techniques in a wireless smart sensor network (WSSN) can eliminate the need to frequently replace batteries, and thus eliminate additional costs to maintain the network. Because energy harvesting has these advantages, the amount of research in the area has been increasing at a tremendous rate in recent years.

The main sources of ambient energy are sunlight, thermal gradient, human motion, vibration, and acoustic noise. Multiple articles reviewing the prospects of ambient energy sources for harvesting can be found in the literature (Glynne-Jones and White 2001; Roundy et al. 2004; Qiwai et al. 2004; Mateu and Moll 2005; Paradiso and Starner 2005). Roundy et al. (2004) compare the power density of available harvesting technologies, as shown in Table 2.6. The authors conclude that an energy harvester can provide a better solution than using batteries alone, especially for systems whose desired lifetime is longer than one year. Paradiso and Starner (2005) made a similar comparison ambient energy sources, shown in Table 2.7, but they focus on slightly different sources. Glynne-Jones and White (2001), Qiwai et al. (2004), and Mateu and Moll (2005) all provide the basic principles associated with energy harvesting techniques, including piezoelectric, electrostatic, and thermal energy. The reports all suggested combining the use of multiple energy harvesting techniques for one application to increase the overall harvesting efficacy in various situations. From the numerous reviews of energy harvesting technologies, it is evident that a harvester may be integrated into a structural health monitoring (SHM) system to provide a near unlimited supply of power.

Table 2.6 Power Densities of Harvesting Technologies (Roundy et al. 2004).

Harvesting Technology	Power density
Solar cells (outdoors at noon)	15 mW/cm^2
Piezoelectric (shoe inserts)	330$\mu W/cm^3$
Vibration (small microwave oven)	116$\mu W/cm^3$
Thermoelectric (10^{o}C gradient)	40$\mu W/cm^3$
Acoustic noise (100dB)	960nW/cm^3

Table 2.7 Energy Harvesting Demonstrated Capabilities (Paradiso and Starner 2005).

Energy Source	Performance
Ambient radio frequency	<1 μW/cm^2
Ambient Light	100 mW/cm^2 (directed toward bright sun)
	100 μW/cm^2 (illuminated office)
Thermoelectric	60 μW/cm^2
Vibrational microgenerators	4 μW/cm^3 [human motion (Hz)] 800 μW/cm^3 [machines (kHz)]
Ambient airflow	1 mW/cm^2
Push buttons	50 μJ/cm^2
Hand generators	30 W/kg
Heel strike	7 W potentially available (1 cm deflection at 70 kg per 1 Hz walk)

Several conceptual designs of energy harvesters have been proposed for structural health monitoring (SHM) applications. For instance, Pfeifer et al. (2001) assessed the feasibility of using self-powered sensor tags to monitor the health of a structure. In the report, a piezoelectric patch powered a microcontroller used to operate the sensor array, perform local computing, and save the results into a radio frequency (RF) identification tag. The data was stored in nonvolatile memory to be subsequently retrieved by a mobile host. In laboratory

experiments, the piezoelectric harvester delivered enough power to the microcontroller for 17 seconds of operation.

Discenzo et al. (2006) created a self-powered sensor node for monitoring the condition of a pump. The node was able to perform sensing, do local processing, and telemeter the results to a command center node. To self-power the node, a piezoelectric energy harvester was used. The device was mounted to an oil pump, and a piezoelectric beam harvested the energy from the pump vibration. The maximum energy created was 40mW.

Another self-powered system was proposed by James et al. (2004) for condition monitoring applications. The device used low-power accelerometers for sensors, and was powered by a vibration-based electromagnetic generator. The generator could provide a constant output of 2.5 mW. Similar to this self-powered system is a design proposed by Elvin et al. (2002) which was powered by piezoelectric patches. Both of these designs, however, lack any local computing capability; they only transmit the direct sensor readings, a great inhibitor when considering SHM applications.

Inman and Grisso (2006) proposed an autonomous sensor node that combined two energy harvesting technologies: ambient vibration and temperature gradient. It contained a battery charging circuit, local computing capabilities, active sensors, and wireless transmission. The authors reported that the elements would be unobtrusive compared to the system being monitored.

As evidenced in Table 2.6 and Table 2.7, solar energy harvesting is perhaps the most powerful of any harvesting technologies available. Raghunathan et al. (2005) systematically analyzed the components, design choices, and tradeoffs involved in designing a solar energy harvesting module. The report illustrated the importance of power management, and presented an efficient solar harvesting module called the Heliomote. The Heliomote allowed for near-perpetual operation of the Mica2 mote.

Although significant strides have been made in the area of energy harvesting for SHM applications, the field requires further research to devise an appropriate solution for the Imote2 WSS platform. In the case of piezoelectric energy harvesters, the power produced is directly proportional to the vibration frequency. Since the natural vibration frequencies of civil

infrastructure systems are low, the power produced with piezoelectric harvesters (e.g. 40mW) is below that required to maintain operation with the Imote2. In the case of the self-powered systems proposed by James et al. (2004) and Elvin et al. (2002), both designs lack any local computing capability; they only transmit the direct sensor readings, a great inhibitor when considering SHM applications. The research presented in this report seeks to devise an effective energy harvesting system for use with the Imote2 platform.

2.4 Sensor data storage on the Imote2

The Imote2's performance characteristics make it the most suitable commercially available wireless smart sensor (WSS) platform. The enhanced performance of the Imote2 comes at a tradeoff to power consumption, and thus a smart, comprehensive power management strategy is in order. To increase the Imote2's efficacy as an autonomous WSS, however, it is also critical to address the issue of data storage. The current method of collecting data on many Imote2 networks relies on the use of a local node and base station, without the option to keep the data on any remote nodes, potentially resulting in data loss if a power failure occurs or communication is disrupted.

The tool developed as part of the Illinois Structural Health Monitoring Project (ISHMP) to collect sensor data from multiple sensors is called *RemoteSensing*. This tool also provides the basis for most of the distributed SHM applications available from the ISHMP. The *RemoteSensing* application tool employs the use of a state machine to determine the timing and flow of the application across an entire network. This state machine is a way to determine how an application behaves when it is in a particular state. The flow chart in Figure 2.4 illustrates the state machine for *RemoteSensing*, while Table 2.8 summarizes each state and transition associated with *RemoteSensing.*

As seen in the state machine illustration in Figure 2.4, the sensor data collected on remote nodes is sent back to the local node following the completion of a sensing event. This flow of data leaves the network susceptible to data loss in the event of power failure in one of the nodes or communication problems between nodes. The autonomous monitoring capabilities of WSSNs using Imote2s would be enhanced by modifying the state machine to

allow sensor data to be stored on non-volatile memory or remote nodes. This storage technique creates a redundancy that prevents data loss from occurring.

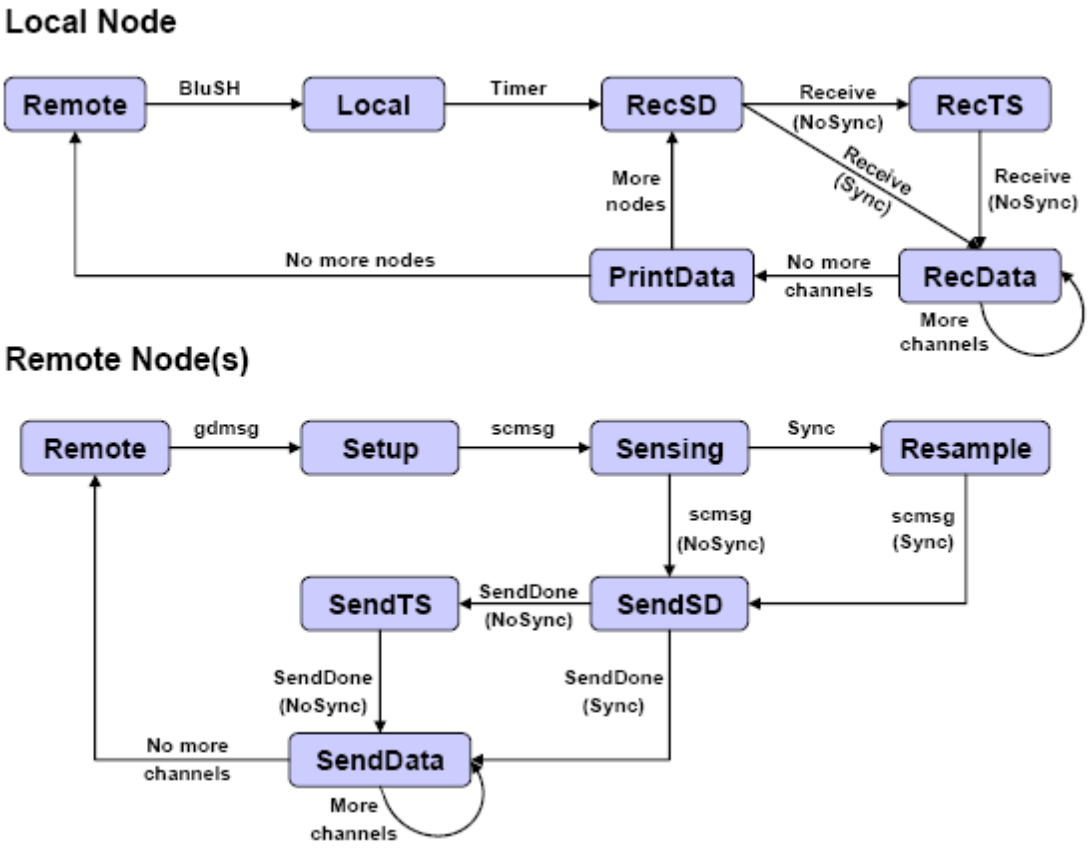


Figure 2.4 *RemoteSensing* state machine for the local node (top) and remote nodes (bottom). Boxes represent states, arrows represent transitions, and arrow labels indicate conditions or actions needed for the transition to occur (Rice and Spencer 2009).

Table 2.8 State and transitions for *RemoteSensing* application (Rice and Spencer 2009).

State	*Description*
Remote	Initial state
Local	Initial local node state
Setup	Receive and store sensing parameters
Sensing	Data acquisition
Resample	Resample of acquired data based on timestamps and initial delay
SendSD	Send sensor data structure
RecSD	Receive sensor data structure
SendTS	Send timestamps (if data is not resampled)
RecTS	Receive timestamps (if data is not resampled)
SendData	Send sensor data
RecData	Receive sensor data
PrintData	Write data to PC
Transition	
BluSH	Application initialized by user through the Blue Shell interface
gdmsg	*GetData* message containing sensing parameters received
Timer	Timer set to wait for remote node(s) to acquire data
scmsg	*StartCollection* or request for data message received
Sync	Resampling flag set
NoSync	Resampling flag not set
sendDone	Previous message sent successfully
receive	Data successfully received

2.5 Topology of a wireless smart sensor network

This research report contains wireless smart sensor terminology that may be unfamiliar to the reader, and thus a brief introduction to the topology of such networks is given here. A WSSN is comprised of three basic units: 1) a base station, 2) a gateway node (sometimes called local node), and 3) leaf nodes (sometimes called remote nodes). The three units are depicted in Figure 2.5.

The base station is typically a data acquisition system on which the sensor data will be permanently stored. The base station also provides the means for users to communicate with nodes in the network. This communication can be done remotely through the internet, or locally using the keyboard of the base station.

The gateway node is connected to the base station, and it transmits all of the sensor data to the base station for permanent storage. This node is used to initiate commands on all leaf nodes, and usually does not perform in sensing. Note that the gateway node is sometimes referred to as the local node, and both terms are used in this report.

The leaf nodes are used to actually perform sensing and computations, and then transmit the data back to the gateway node. Commands on leaf nodes are initiated by the gateway node. These nodes are not in direct communication with the base station. Leaf nodes are also called remote nodes in other literature and in this report.

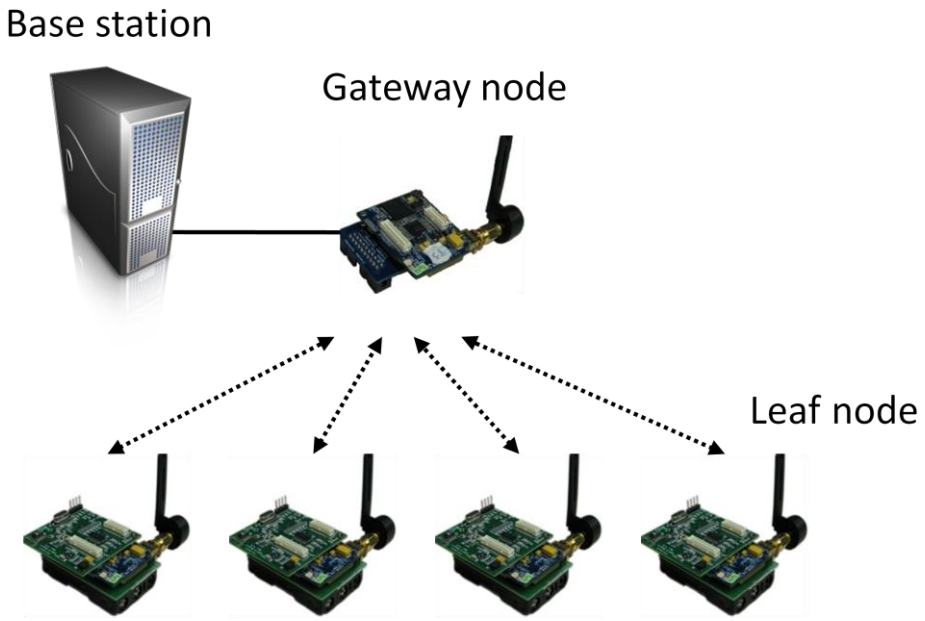


Figure 2.5 Topology of a typical WSSN of Imote2s.

2.6 Summary

The background presented in this chapter reveals the superiority of the Imote2 wireless sensor platform for high sampling rate SHM applications. The Imote2 has been the target of many applications developed for SHM, especially by the Illinois Structural Health Monitoring Project

(ISHMP). Despite its superior performance characteristics, the Imote2's heavy power consumption is a limitation for realizing long-term autonomous WSSNs.

Researchers with the ISHMP have successfully implemented strategies to reduce the power consumption of the Imote2 while running in a WSSN. By utilizing *SnoozeAlarm*, studies show that the Imote2 can operate up to 60 days in a network without requiring a change of batteries. Implementing this power minimization strategy is one side of a smart power management scheme, and to complete the scheme requires a perpetual supply of energy into the system.

Energy harvesting methods represent a promising method to supply perpetual energy into a WSS module and realizing a smart power management scheme. Previous efforts to utilize energy harvesting measures for SHM applications have shown good results, but no such efforts will supply enough power to keep the Imote2 operational. Of the myriad energy harvesting techniques, photovoltaic energy harvesting is one of the most reliable and mature methods. Utilizing photovoltaic panels can eliminate the need for consistent battery replacements now required in WSSNs.

To increase the autonomy of WSSNs, the sensor nodes should be able to utilize non-volatile memory for sensor data storage. Rather than sending data back to a gateway node at the base station, as is traditionally done, storing on local memory can save data in the case of power failure or communication loss.

The research presented in this report seeks to address the problems identified herein by providing a solar energy harvesting system compatible with the Imote2. Additionally, software applications will be created to utilize non-volatile memory and allow for data retrieval even after power failure or communication failure. All contributions made in this report will be open-source and made available to the public to encourage further advancements in these areas, and ultimately improve the effectiveness of autonomous monitoring for SHM applications.

Chapter 3 Solar Energy Harvesting

Solar energy harvesting using photovoltaic cells represents one of the most mature and reliable techniques available for utilizing ambient energy in a wireless smart sensor network (WSSN). As described in section 2.3, it has the potential to produce 100 mW/cm^2 of power, far more than most harvesting technologies. With the large energy demands of high-performing wireless smart sensor (WSS) platforms such as the Imote2, solar energy harvesting is an ideal way to provide power during the duration in which a WSSN is in operation. First, an overview of the technology behind solar cells is given, followed by a discussion of the variation in types of solar cells. Next, a description of various rechargeable battery chemistries is given, including advantages and disadvantages of each type. Ultimately, the reader will gain a solid understanding of the various components involved in solar energy harvesting.

3.1 Photovoltaic cells

In brief, photovoltaic (PV) cells – commonly called solar cells – convert the sun's energy into usable electricity. Photovoltaic cells are made of materials known as semiconductors like silicon, the most commonly used PV semiconductor. Silicon has only four valence electrons in its outer shell, where it desires to have a full eight. Silicon atoms thus share electrons with other silicon atoms to form a crystal. In PV cells, the silicon is mixed with impurities like phosphorous atoms, which have five valence electrons – one more than silicon. The extra phosphorous electron isn't bonded to a neighboring atom, and is easy to break free. When the photons from the sun strike the PV cell, they are partially absorbed into the semiconductor material, transferring energy from the light to the semiconductor. The energy loosens electrons and allows them to flow freely. Half of the PV cell is mixed with phosphorous or similar negative atoms, and half is mixed with boron – with three valence electrons. The negative and positive type silicon combined together forms an electric field. Each PV cell has at least one electric field that forces the freed electrons to flow in a certain direction. This flow of electrons is called current. Placing metal contacts on the top and bottom of a PV cell allows the current to be drawn off and used externally. The current, couple with the cell's voltage, define the power that the PV cell can produce. The operation of a PV cell is illustrated in Figure 3.1.

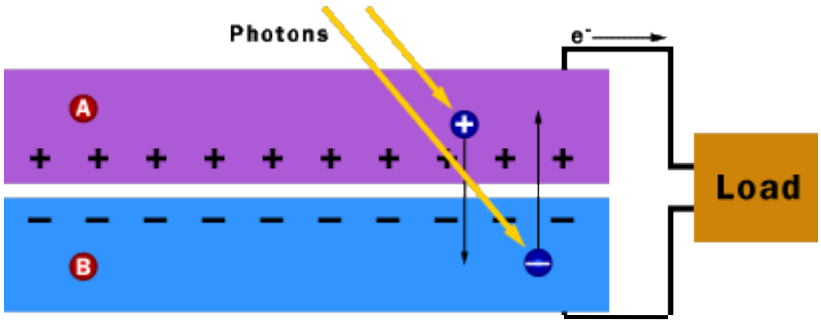


Figure 3.1 Operation of a PV cell (Aldous 2000).

The basic construction of PV panels is based around the PV cell. PV cells are often connected together and encapsulated in one module, or panel. The modules commonly include a sheet of glass on the sun side to allow light to penetrate while still protecting the actual semiconductor cells from damage. The cells within a module are normally connected in series to increase the overall voltage of the module. (To prevent current flow from a producing cell into a dead or shaded cell, many PV modules utilize bypass diodes, effectively allowing at least partial power output in shade.) The modules can then be connected in parallel to increase the output current, if desired. To prevent electricity from flowing back into the solar module, a blocking diode is commonly used between the module output and its target input. While some PV modules may vary, this type of construction is most common. An illustration of this construction is found in Figure 3.2.

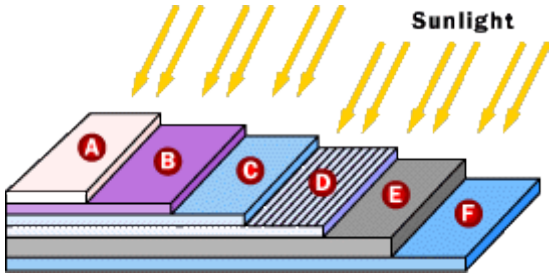


Figure 3.2 Basic structure of a generic silicon PV cell (Aldous 2000).

In addition to the basic construction of PV modules, it may also be helpful to understand the difference between cells two prominent cell types: monocrystalline and polycrystalline. Monocrystalline cells are cut from silicon that was derived from a single crystal, whereas polycrystalline cells are cut from multifaceted crystalline material, meaning it grows in multiple directions. Traditionally, monocrystalline cells have been more efficient at converting sunlight into usable electricity, but recent developments in polycrystalline technology have closed the

performance gap between the two. The easiest way to identify a polycrystalline cell is that it will exhibit a shattered appearance, whereas the monocrystalline cell looks homogeneous. This is illustrated in Figure 3.3.

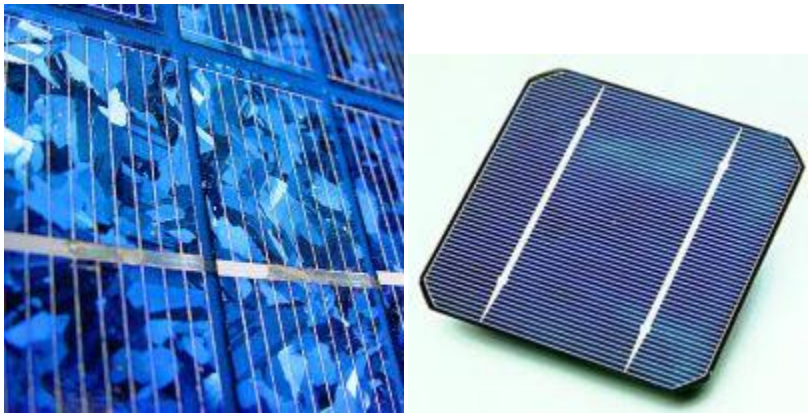


Figure 3.3 Polycrystalline PV cell (left) and monocrystalline PV cell (right).

When considering the use of solar PV panels for use in WSSNs, it is crucial to understand their voltage and current characteristics. Raghunathan et al. (2005) performed a study to characterize a 3.75” x 2.5” panel from Solar World Inc., the 4-4.0-100. This panel is typical of the commercially available panels, and the results of the study can be extended to similar panels. In the study, the authors noted that panels are characterized by two parameters, the open circuit voltage and the short circuit current. These two parameters are found on the x- and y- intercepts of the V-I curve shown in Figure 3.4. From Figure 3.4, a few observations are made. First, it is evident that the solar panel behaves as a voltage limited current source. Additionally, an optimal operating point exists, meaning that power output from the panel is maximized at that point. Finally, as the amount of solar radiation decreases, the short circuit current decreases; however, the open circuit voltage remains nearly constant (Raghunathan et al. 2005). Understanding these characteristics is beneficial in achieving an effective solar energy harvesting system.

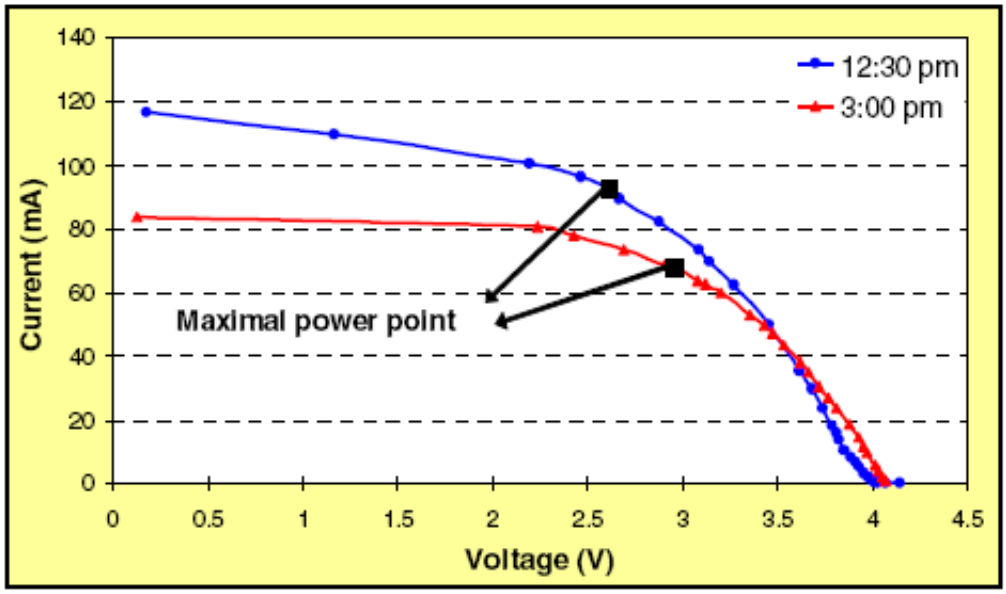


Figure 3.4 Measured V-I characteristics of the Solar World 4-4.0-100 solar panel (Raghunathan et al. 2005).

Though solar energy harvesting with PV cells represents a reliable, promising method for powering WSSNs performing SHM, there are some shortcomings of solar harvesting that need to be addressed. For instance, it is clear that the sun will not shine on solar panels 24 hours a day for the entire year. PV panels also act as current sources, making it difficult to supply a constant voltage. Thus, despite the power output capable with solar panels, they are not capable of supplying power directly to the WSS modules at all times. To remedy this issue, an energy storage device is needed. Perhaps the best known and most mature energy storage device is the battery.

3.2 Rechargeable batteries

Rechargeable batteries represent a mature, reliable energy storage technology. To increase the effectiveness of solar energy harvesting, batteries can be used to store the energy from the solar panel, and subsequently to power the wireless smart sensor (WSS) module. In 2007, Casciati and Rossi listed three requirements of a battery for use in an energy harvesting system, which are:

1) The battery must be rechargeable;

2) The battery must present a high energy density;
3) The battery must show a low self-discharge rate, so that it can survive long periods without supply.

Items 2 and 3 in the prior list of requirements are becoming ever more crucial in light of the current performance, and hence energy consumption, of today's WSSs. Unfortunately, the world is still waiting on a technological breakthrough in battery performance. The capacity of batteries has improved slowly in the past 30 years, whereas computational capabilities have dramatically increased, as illustrated in Figure 3.5. Despite the lagging performance of batteries, however, the current rechargeable chemistries are sufficient for energy harvesting systems if selected appropriately.

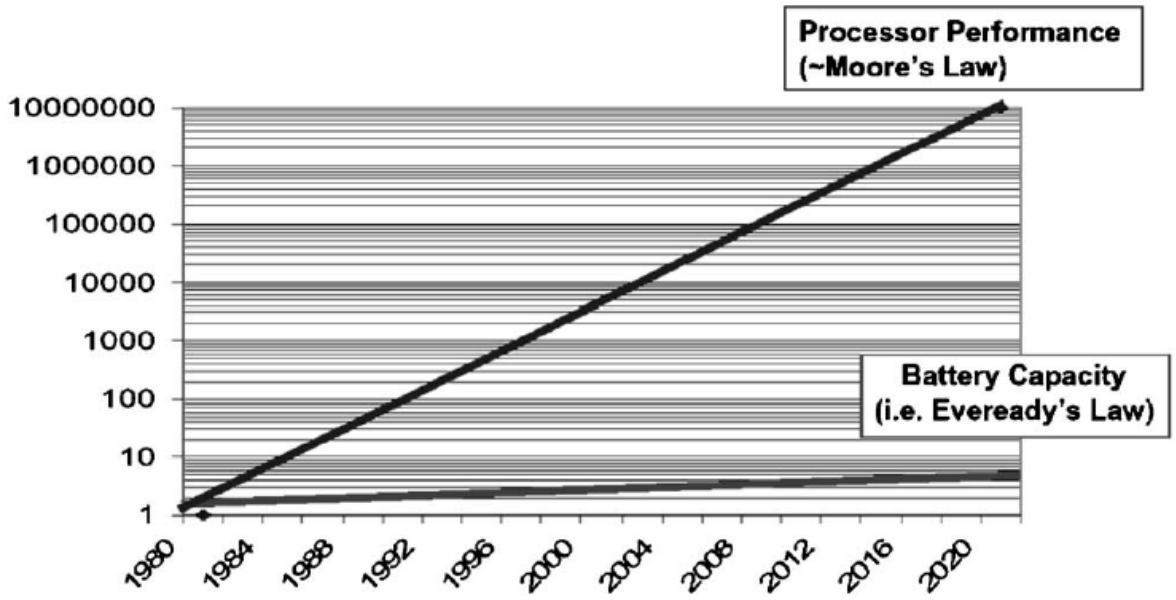


Figure 3.5 Battery capacity versus processor performance (Park et al. 2008).

Five primary rechargeable battery types exist in today's market. They are Nickel-Cadmium (Ni-Cd), Nickel metal hydride (Ni-MH), sealed lead acid (SLA), Lithium-ions (Li-ions), and polymer lithium-ions (Polym.-Li or Li-poly). Note that Lithium-ions and polymer lithium-ions are often grouped together because they exhibit the same or similar behavior, and they will be referred to as Li+. Of these five types, SLA and Ni-Cd batteries are seldom used in energy harvesting systems. The reason SLA batteries are seldom used is because they have relatively

low energy density, while Ni-Cd batteries are rarely used because they suffer from capacity loss caused by shallow discharge cycles, often called the memory effect. Forgoing the use of SLA and Ni-Cd batteries leaves a choice between Ni-MH and Li+ batteries. The choice is not entirely generalized, as there are advantages to each type. Li+ batteries are more efficient, have a longer cycle lifetime, and exhibit a lower self-discharge rate. They are, however, more expensive than Ni-MH batteries. The properties of these five battery types are summarized in Table 3.1.

Table 3.1 Properties of five classes of existing rechargeable batteries (Casciati and Rossi 2007).

Cell type	Ni-Cd	Ni-MH	SLA	Li-ions	Polym.-Li
Energy density (Wh/Kg)	50	75	30	100	175
Life cycle (charges-discharges)	1500	500	200-300	300-700	600
Self-discharge (charge % at time)	60% 4 months	15% 1 month	60% 24 months	40% 5 months	8% 1 month
Nominal voltage (V)	1.25	1.25	2	3.6	2.7

In addition to the characteristics summarized in Table 3.1, it is also beneficial to know the run time of any particular battery. To select an appropriate battery, the conditions under which battery cells are rated should be understood. The standard rating of a cell battery if abbreviated as *C*, and is the capacity of a new cell subjected to constant-current discharge at room temperature. The capacity, however, may actually vary inversely with discharge rate, so capacity ratings depend on the discharge rate used. As an example, Figure 3.6 illustrates that the effective capacity of a lithium-ion cell is decreased as the discharge rate is increased. This phenomenon is called the capacity offset, and is exhibited by most cell chemistries. When considering the use of batteries in SHM applications, this capacity offset is satisfying; the

relatively low current consumption of the WSS modules (tens of mA) suggests that the capacities realized from the batteries will exceed their rated capacities.

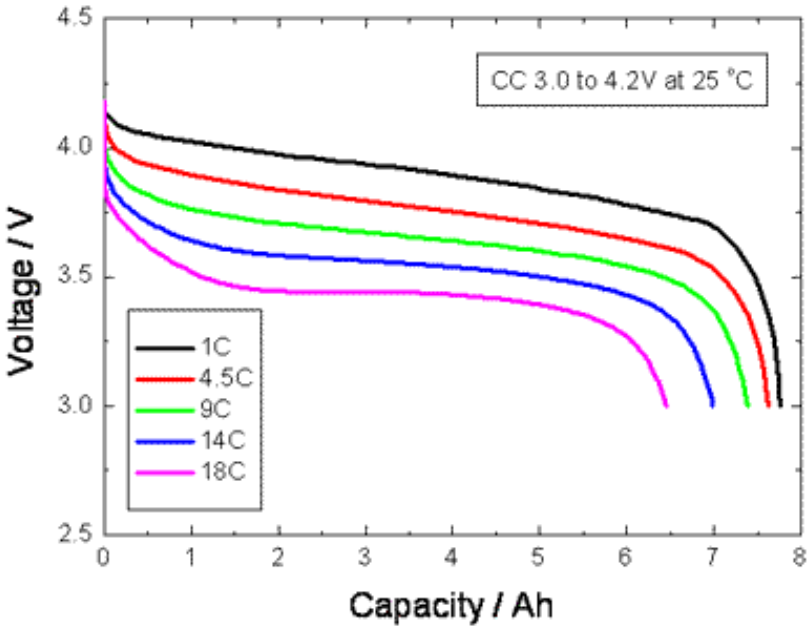


Figure 3.6 Capacity offset of Li-ion cell discharged at various rates (Woodbank 2005).

As a means to further assess the suitability of certain battery cell chemistries for use in energy harvesting systems, it is useful to know the fashion in which each cell type discharges. This discharge information is useful because some WSS platforms will not perform reliably if the power supply voltage diminishes below a certain point. For example, the Imote2 equipped with the SHM-A sensor board will not operate if the voltage is below 3.8 V. The discharge curves shown in Figure 3.7 provide a means of predicting the voltage of each battery type over its cycle time. Of particular interest is the fact that the Ni-MH and Li+ batteries yield a fairly constant voltage during their discharge time. They begin to drop in voltage only after about 85 percent of the battery has been discharged. These discharge characteristics are ideal when the load requires a high voltage.

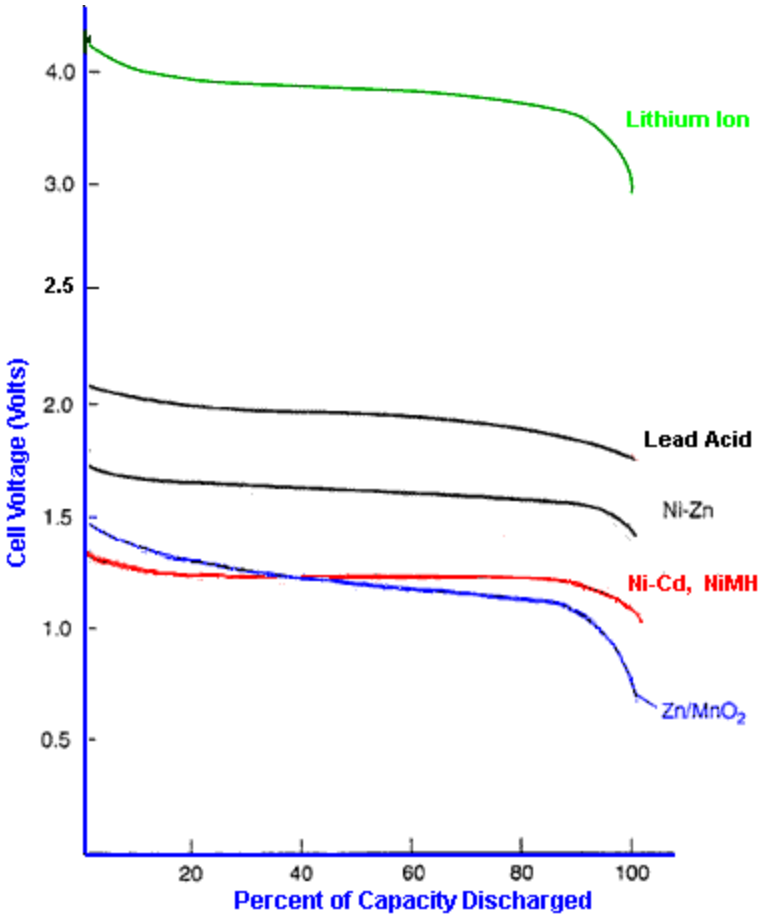


Figure 3.7 Discharge curves of various cell chemistries (Woodbank 2005).

3.3 Summary

Utilizing solar energy harvesting can be a reliable, effective way to provide power to WSSNs. Solar photovoltaic panels can produce enough electricity to recharge batteries, and they represent a mature technology that has proven effective, making them ideal for WSS platforms such as the Imote2.

Rechargeable battery performance has not kept up with the performance of electronics, but they can still be utilized to supply ample amounts of power if chosen appropriately. Lithium-polymer batteries provide many advantages over other chemistries, such as high energy density, low self-discharge rate, long cycle lifetime, and an ideal discharge curve. With all of these advantages, Lithium-polymer batteries are the ideal chemistry choice for use in an energy harvesting system.

Combining the use of photovoltaic panels and rechargeable batteries can result in a perpetual energy supply for WSSNs. With the appropriate selection of panels and batteries, WSSN operation can continue without the need for swapping batteries, effectively increasing the autonomy of these networks.

Chapter 4 Implementation of solar energy harvesting on the Imote2

The Imote2 is the ideal wireless smart sensor (WSS) platform to take advantage of energy harvesting technology. With its advanced computational ability, the Imote2 is perfectly suited for many structural health monitoring (SHM) applications requiring high sampling rates. To compensate for its relatively high power consumption, a smart power management scheme needs to be implemented on the Imote2. One critical aspect of this power management scheme is perpetuating the power supply through energy harvesting strategies. With solar energy harvesting representing a mature, reliable, and powerful technique, it is an ideal choice for SHM applications with the Imote2. This chapter will detail the process of implementing a solar energy harvesting solution on the Imote2. First, a briefing on the selection of solar panel and rechargeable battery is given. Next, an overview is given regarding the hardware modifications necessary to realize a successful solution. Finally, the software used to recharge the battery is discussed, including charging parameters. The goal of this chapter is to detail all of the steps involved in implementing a proficient solar energy harvester on the Imote2.

4.1 Solar panel selection

In determining the proper photovoltaic (PV) solar panel for an energy harvesting system, the panel characteristics outlined in section 3.1 must be considered. In particular, the energy demands of the system should be known in order to select a panel that will yield its optimal power output. Additionally, the operating conditions of the wireless smart sensor network (WSSN) must be addressed, for the sizing of the solar panel heavily depends on the amount of sunlight in the operating area. Furthermore, WSS modules limit the acceptable voltage range for external sources, so knowing this range is essential in preventing damage occurrence in the system. In the case of the Imote2, all of these issues are discussed in this section, and a solar panel is selected for full scale testing.

As mentioned in section 2.2, great efforts have been made to reduce the energy demands of the Imote2 for SHM applications; the average current draw is estimated to be under 15mA for the parameters listed in Table 2.4 (Rice and Spencer, 2009). Knowing the average current draw is the first step in determining a proper solar panel for the energy

harvesting system. The next step is to consider the conditions in which the WSSN will be operating.

With the energy consumption of the system estimated, considerations need to be made regarding the operating conditions of the network. In particular, the number amount of sunlight available in the network's region is a crucial parameter. To assist with determining the sunlight, the number of equivalent sun hours is published for regions around the world. Equivalent sun hours are the number of hours one can expect peak production from the solar panel. This research, as part of the Illinois Structural Health Monitoring Project (ISHMP), was conducted at the University of Illinois in Urbana-Champaign, and the number of sun hours for this region averages around 3.14 hours per day (Advanced Energy Group).

Knowing the equivalent sun hours and the energy demands of the system, one can use a series of simple equations to approximate the minimum necessary PV panel dimensions. These are as follows:

$$\boldsymbol{Daily\ Consumption}\ (\boldsymbol{mA\!\not h/\!\not day}) = \boldsymbol{Current\ Draw}\ (\boldsymbol{mA}) * \boldsymbol{24\!\not h} \qquad \textbf{(Eq. 4.1)}$$

$$\boldsymbol{Required\ Solar\ Panel\ Current}\ (\boldsymbol{mA}) = \frac{\boldsymbol{Daily\ Consumption}\ (\boldsymbol{mA\!\not h/\!\not day})}{\boldsymbol{Avg.Sun\ Hours}\ (\boldsymbol{\not h/\!\not day})} \qquad \textbf{(Eq. 4.2)}$$

Following Eq. 4.1 and Eq. 4.2 will yield a net charge of the battery used for energy storage (which is discussed in section 4.2). Note that the daily consumption is given in units of mAh to remain consistent with the capacity units for batteries. Following Eq. 4.1 and Eq. 4.2 for a region in Illinois yields a required solar panel current output above 115mA. With the minimum current output established, the voltage range of the solar panel needs to be established.

To establish the voltage range for a solar panel, consideration must be given to the components on the WSS module. The Imote2 is equipped with a power management integrated circuit (PMIC) to control charging of batteries, as well as to supply the processor with all the required voltage domains (Crossbow 2007a). The PMIC model is the DA9030 from Dialog Semiconductor, which allows programmable charging from 40mA to 1400mA. In addition to the charging current range, the DA9030 accepts input voltages for charging purposes between 4.6V

and 10V (Dialog Semiconductor, 2005). The characteristics for a solar panel are summarized in Table 4.1.

Table 4.1 Solar Panel Characteristics for use with the Imote2 in Illinois.

Output Voltage Range (V)	4.6 - 10
Output Current Range (mA)	115 - 1400

With the criterion established in Table 4.1, an appropriate solar panel can be selected for use in a solar energy harvesting system on the Imote2. Beyond the criterion in Table 4.1, it is also beneficial to consider the overall dimensions of the panel, as well as its cost. To maintain the price savings realized with WSSs (over traditional wired systems), the cost of the solar panel must be considered, and to keep the WSSs inconspicuous, the dimensions of the solar panel should be minimized. For these reasons, the Solarworld SPE-350-6 is selected for use in the ISHMP. The SPE-350-6 has a short circuit current of 350mA and an open circuit voltage of 9V (Solarworld, 2009). An image of this panel is shown in Figure 4.1.

Figure 4.1 Solar panel (SPE-350-6) for the ISHMP.

4.2 Rechargeable battery selection

As mentioned in section 3.1, solar panels cannot be used effectively to directly power a wireless smart sensor (WSS) module, and thus should be used to supply energy to an energy storage device like a battery. As mentioned in section 4.1, the Imote2 is equipped with a PMIC to handle charging of batteries. The PMIC supports single cell Li-ion batteries at 4.1V and 4.2V, as

well as Li-polymer packs (Crossbow 2007a). Selection of the appropriate battery for the Imote2, therefore, is narrowed to these chemistry types.

The merits of the Li+ type batteries were discussed in section 3.2. In brief, they are energy dense, they do not suffer from memory effects, they have a large cycle life, and they self-discharge at a low rate. These characteristics make Li+ batteries ideal for use in an energy harvesting system.

In selecting the battery, it is evident that the chemistry type will be Li+; however, further considerations need to be given to things such as capacity, voltage rating, and built-in protection circuitry. It is ideal to select a battery with a high capacity to keep the system running if the average sun levels required for charging are not met on a particular day. The battery should have a protection circuit built in to prevent over discharge or over charge. For these reasons, the Li-polymer battery called the Powerizer from batteryspace.com is selected for the ISHMP. Its capacity is 10,000mAh, its nominal voltage is 3.7V (AKA 4.1V in some rating systems), and it contains a protection circuit to prevent damage from over discharging or charging. This battery is shown in Figure 4.2.

Figure 4.2 Rechargeable battery, Powerizer.

4.3 Hardware modifications

While the PMIC can be used to charge batteries on the Imote2, hardware modifications must be made to the system to enable charging. If a solar panel is applied as an external power source without making the proper modifications, the Imote2 may be damaged. With the proper modifications, the nCHARGE_EN pin on the Imote2 will be pulled to a position that routs the

USB input to the Vchg pin of the PMIC, effectively activating the charging mode. The diagram in Figure 4.3 reveals how the power is routed when the nCHARGE_EN pin is pulled into the low position.

To enable active charging on the Imote2, two modifications need to be made to the Crossbow battery board (IBB2400). (These modifications will not enable charging with the Intel battery board; only the Crossbow battery board should be used for this energy harvesting system.) The two necessary modifications are as follows:

1) Populate position R3 with a 0 ohm resistor.
2) Populate position R1 with a 0 ohm resistor.

The first modification will connect the nCHARGE_EN pin to ground, which enables charging by rerouting the power as described earlier. The second modification bypasses the polarization protection diode on the battery board. Without bypassing this diode, current is not able to flow back into the battery, a necessary process for charging. A picture showing the locations of R1 and R3 on the battery board is seen in Figure 4.4.

When using the Li+ batteries with the energy harvesting system, the battery should be directly connected to the battery board. The battery should be connected to the (+) and (-) terminals on the battery board, as seen in Figure 4.4. Also, when connecting the solar panel to the Imote2, the lead wires from the panel can be outfitted with a Mini-USB 5-pin male connector, which will fit in the female connector on the Imote2. The positive (+) wire should be connected to pin 1, while the negative (-) wire should connect to pin 4, the ground pin. The remaining three pins will remain unconnected. A diagram of the Mini-USB connector is shown in Figure 4.5 for pin references.

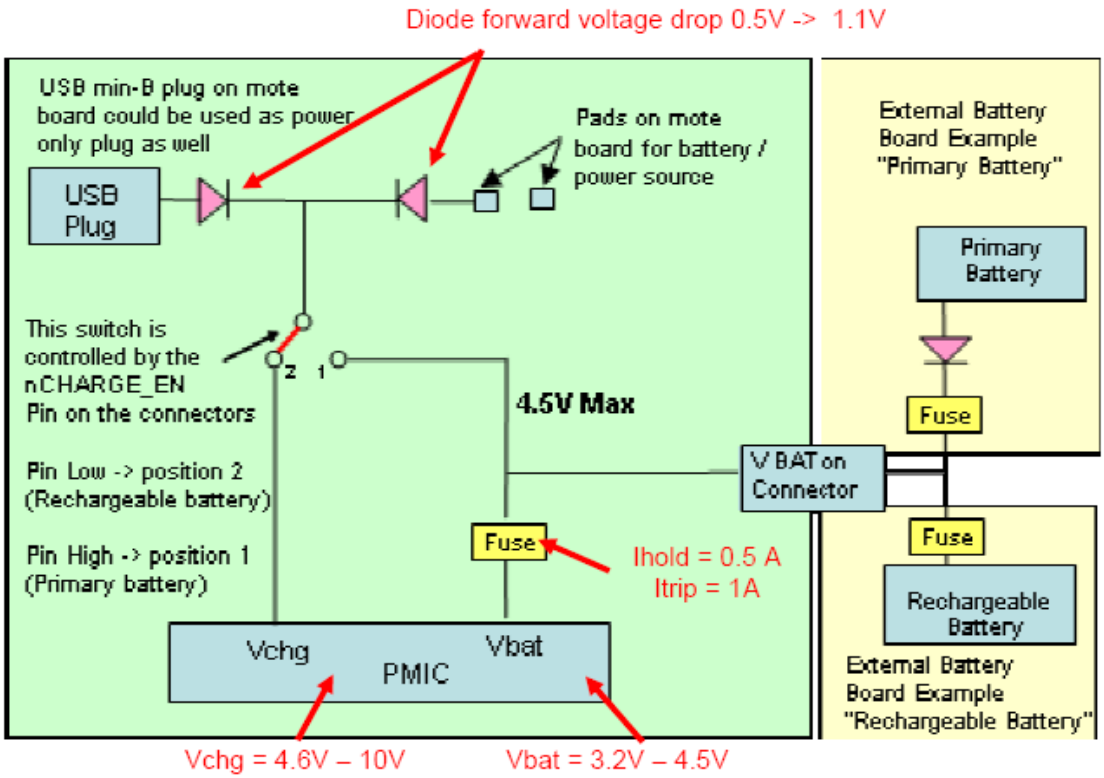


Figure 4.3 Power supply options for the Imote2 (Crossbow 2007a).

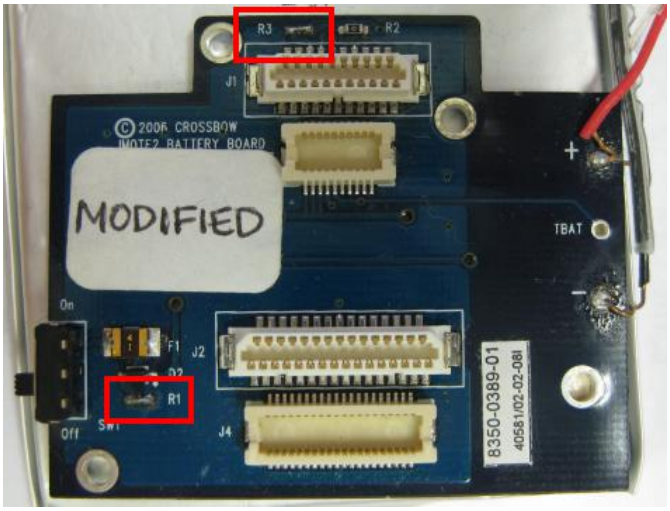


Figure 4.4 Battery board modifications for charging (R1 and R3 locations).

Figure 4.5 Mini-USB pins.

With all of the necessary hardware modifications in place, the system is set up for energy harvesting. The nCHARGE_EN pin will be pulled low with the battery board attached, and the protection diode will be bypassed to allow current to flow into the battery. The energy harvesting system will not function appropriately, however, without proper software control, described in the next section.

4.4 Charging Software

The DA9030 PMIC from Dialog Semiconductors interfaces directly with Lithium battery packs, and can handle an unregulated supply up to 10 volts. The charger supports constant current and constant voltage charging modes, as well as trickle charging. With the features of the PMIC combined with software unique to the Imote2, an effective charging scheme can be developed.

The charging block of the PMIC contains the following features (Dialog Semiconductor, 2005):

- Accurate current regulation, which is needed for constant current charging. Programmable from 40 to 1400mA in 100mA steps.
- Accurate voltage regulation, which is needed for constant voltage charging. Programmable from 4.0 to 4.35V in 50mV steps.
- Current monitoring, which is always active when the charger is on.
- Over voltage monitoring, which is always active when the charger is on.
- Temperature monitoring, which is needed to turn off charging when the battery temperature is too high or too low.

To elaborate on the first and second bullets, Lithium type batteries undergo a unique charging scheme when charged properly; they start in a constant current mode until the voltage of the cell is near 4.0 volts, and then they undergo charging in a constant voltage mode. Included in the ISHMP tool suite available at http://shm.cs.uiuc.edu/software.html is a module called PMICM.nc written by Lama Nachman and Robert Adler of the Intel Corporation. This module contains the code to control the charging of the Imote2 with an energy harvesting system. In this module, the current for the constant current mode can be set, and the voltage can also be set. These values are changed by changing the value of PMIC_CC_ISET() and PMIC_CC_VSET(), for current and voltage, respectively. For the ISHMP, the current is set to 200mA – well above

the required 115mA for net charging – and the voltage is set to 4.2V – the necessary value to reach peak charge.

Though the charging parameters are set in PMICM.nc, further code is necessary to make the energy harvesting system function properly alongside *SnoozeAlarm* – the energy saving feature incorporated into many sensing applications available from the ISHMP. The module to allow proper charging with *SnoozeAlarm* is called ChargerControlM.nc, and was developed as part of the research for this report. With the ChargerControl program enabled, the Imote2 will check the supply voltage each time it awakes from sleep, and will initiate charging if that supply voltage is adequate. ChargerControl also checks the battery voltage, and only initiates charging if the voltage is below a specified value, such as 4.0V. A flow chart of these operations is shown in Figure 4.6, and the actual code is shown as a screen shot in Figure 4.7. To enable ChargerControl, the following line must be included in the Makefile for RemoteSensing or other sensing application:

```
PFLAGS += -DENABLE_CHARGER_CONTROL
```

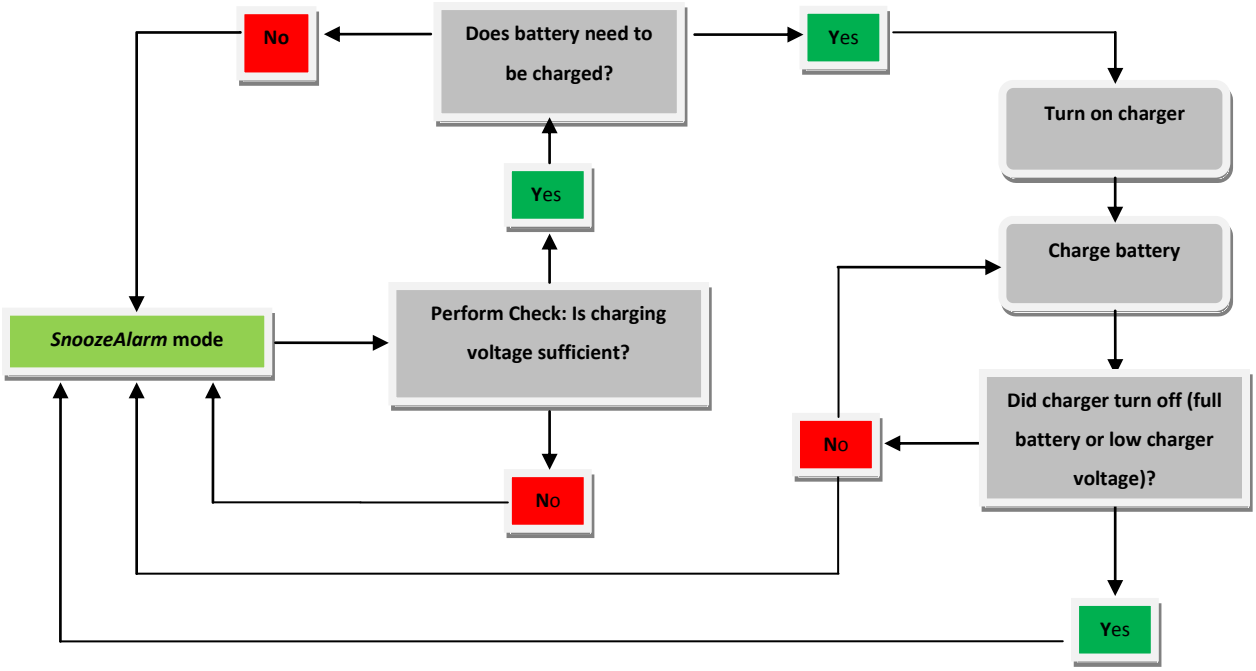


Figure 4.6 Flow chart of *ChargerControl* operation.

```
module ChargerControlM
{
    provides {
        interface StdControl;
        command void setNoInterruptFlag();
    }
    uses {
        interface PMIC;
        interface SnoozeAlarm;
        interface Sleep;
        interface Timer;
    }
}

implementation
{
#include "pmic.h"
#include "FlashConstants.h"
#define CHARGER_CHECK_INTERVAL (30*1000) //30 seconds

    uint8_t sleepFlag, noInterruptFlag;

    command result_t StdControl.init()
    {
        return SUCCESS;
    }

    task void checkChargerTask()
    {
        uint8_t cval, bval;
        call PMIC.getChargerVoltage(&cval);
        call PMIC.getBatteryVoltage(&bval);
        trace(DBG_USR1, "Charger voltage = %.3f, battery voltage = %.3f.\r\n",
            CHARGER_VOLTAGE(cval), BATTERY_VOLTAGE(bval));
        trace(DBG_USR1,          "CHARGER_VOLTAGE_TARGET          =          %f,
CHARGER_TURN_ON_BATTERY_VOLTAGE = %f\r\n",
            CHARGER_VOLTAGE(CHARGER_VOLTAGE_TARGET),
CHARGER_TURN_ON_BATTERY_VOLTAGE);
        if(cval    >    CHARGER_VOLTAGE_TARGET    &&    BATTERY_VOLTAGE(bval)    <
CHARGER_TURN_ON_BATTERY_VOLTAGE) {
            call PMIC.enableCharging(TRUE);
        }
    }

    command result_t StdControl.start()
    {
#ifdef ENABLE_CHARGER_CONTROL
        post checkChargerTask();
#endif
        return SUCCESS;
    }
```

Figure 4.7 Code for *ChargerControl* to initiate charging.

4.5 Summary

The Imote2 platform is well suited to take advantage of solar energy harvesting technology. The implementation discussed in this section details the selection of a solar panel and rechargeable battery for use with the Imote2 to create an energy harvesting system. The implementation of the energy harvester allows perpetual power supply to the sensor nodes, effectively completing the smart power management strategy.

Chapter 5 Non-volatile memory storage

Increasing the autonomy of wireless smart sensor networks (WSSNs) can be achieved by utilizing non-volatile memory for sensor data storage. As mentioned in section 2.4, the traditional sensing scheme for applications in the ISHMP Toolsuite involves sending the sensed data back to a base station node immediately following the completion of a sensing event. Problems arise, however, if battery power is limited, if communication problems are experienced in the network, or if a base station is unavailable. To mitigate these problems, and increase the autonomy of WSSNs, sensor nodes should have the option of storing their sensed data in local, non-volatile, flash memory. This chapter describes an application developed for this research that achieves exactly that; it stores sensed data to flash memory on the Imote2. First, an overview of a helpful application called *SensingUnit* is described. Next, the newly developed application, *RemoteFlashSensing*, is described, followed by directions to operate the application. The goal of this chapter is to provide the reader with insight into the flash storage option on the Imote2.

5.1 *SensingUnit*

SensingUnit is a service component in the ISHMP Services Toolsuite that performs synchronized or unsynchronized sensing and outputs the measured data. Full details of this service are given in Sim and Spencer (2009), but pertinent details are included herein for completeness. *SensingUnit* is located in the directory $SHMROOT/tools/SensingUnit in the ISHMP Services Toolsuite.

Because sensing is required in all SHM applications, *SensingUnit* is a good starting point for the development of new applications. In the case of storing the sensor data into flash memory, *SensingUnit* can be used to output the measured data into an application that organizes the data and stores it into flash memory.

5.1.1 *SensingUnit* state machine

In similar fashion to *RemoteSensing*, a state machine is utilized to control the timing and flow in *SensingUnit.* States are defined based on the task of each node in a WSSN using *SensingUnit*,

and transitions between states are triggered by events such as the completion of sensing. For instance, the default state of a leaf node is “DISABLED”, which is changed to “INIT” in the initialization, and then to “SYNC” when the time synchronization commences. The state is changed to “SENSING” when the synchronization is complete. While in the “SENSING” state, the leaf node measures data. This process is illustrated in Figure 5.1 for both the leaf and gateway nodes.

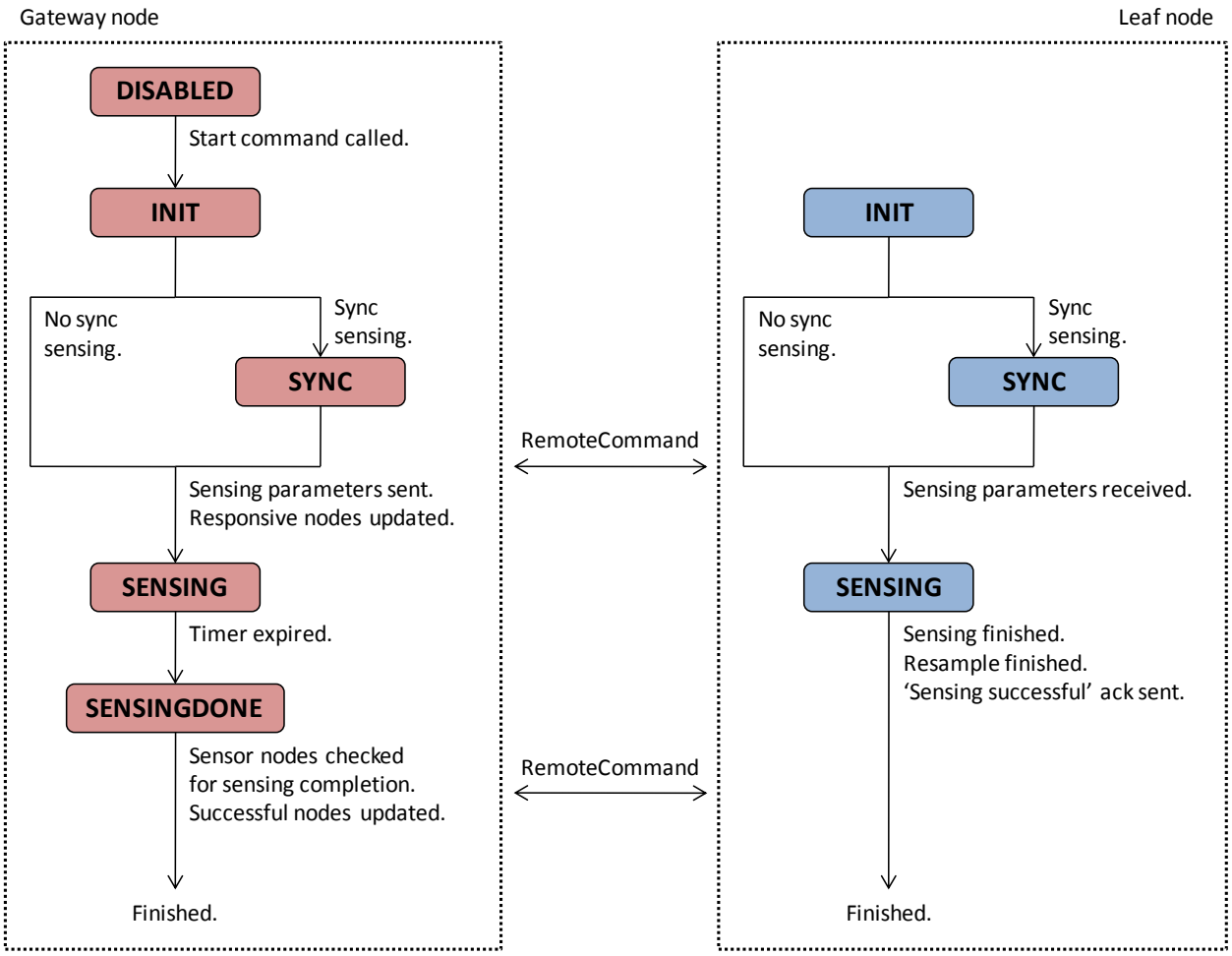


Figure 5.1 Flowchart of *SensingUnit* (Sim and Spencer 2009).

SensingUnit provides an interface so that it may be used as a component in higher-level applications. The higher-level application utilizing *SensingUnit* calls a command to perform a specific task. Subsequently, and event handler is triggered when a specific event takes place. The commands and event handlers provided by the *SensingUnit* interface are as follows (Sim and Spencer 2009):

Interface for gateway node:

- Command “setNodes” tells *SensingUnit* which sensor nodes are supposed to measure data
- Command “setParameter” configures the parameters for sensing (sensing channel information, the length of data, sampling rate, and an option for synchronized sensing)
- Command “startSensing” causes *SensingUnit* to start sensing based on the previously set parameters
- Command “stop” terminates the operation of *SensingUnit* in the gateway node
- Event handler “done” is triggered when *SensingUnit* completes its task

Interface for leaf node:

- Command “init” initializes *SensingUnit*
- Command “stop” terminates the operation of *SensingUnit* in the leaf node
- Event handler “done” is triggered when *SensingUnit* completes its task

In an application that uses *SensingUnit*, the commands “setNodes”, “setParameters”, and “startSensing” are called sequentially to begin sensing, and event handler “done” is triggered at the completion of *SensingUnit*. Thus, *SensingUnit* can be easily embedded in higher-level applications. An illustration of how to interface to *SensingUnit* with a higher-level application is shown in Figure 5.2.

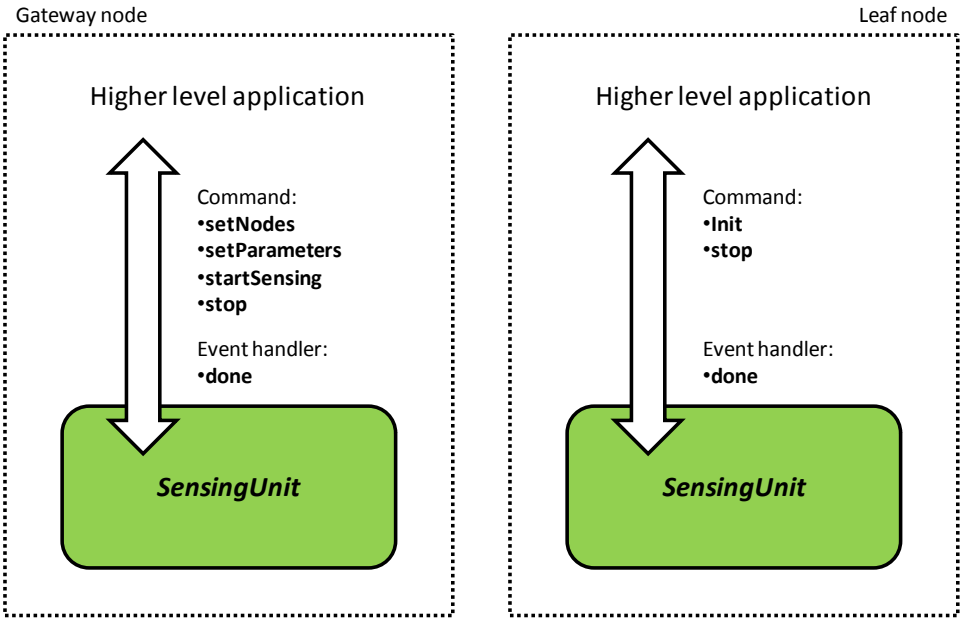


Figure 5.2 Schematic view of the *SensingUnit* interface (Sim and Spencer 2009).

5.2 *RemoteFlashSensing*

RemoteFlashSensing is an application developed for the ISHMP Toolsuite that utilizes *SensingUnit*. *SensingUnit* provides the interfaces for specifying necessary parameters and accessing sensor data, and *RemoteFlashSensing* stores the sensor data to non-volatile, flash memory. By storing the measured data in flash memory, it remains available for retrieval at any time in the future. This is especially beneficial if the remote nodes experience troubles in communicating back to the local node; rather than losing the data in light of communication issues, it remains on the leaf node for later retrieval. This section describes how *RemoteFlashSensing* works, and gives directions on how to run the application.

5.2.1 How *RemoteFlashSensing* Works

RemoteFlashSensing performs two primary functions while operating in a WSSN: 1) It writes the sensor data into flash memory after sensing and 2) it reads the sensor data back from flash memory when instructed to do so.

Writing into flash

Taking advantage of the *SensingUnit* interfaces, *RemoteFlashSensing* is supplied with a sensor data structure upon the completion of a sensing event. The measured sensor data is actually located within multiple channel buffers, and so the sensor data structure contains pointers to

the locations of each channel buffer. To write the measured data into flash memory, however, all of the data must be contained in one long buffer. This buffer must include the sensor data structure, the measured sensor data, and the data timestamps if the data was not resampled. To facilitate the temporary storage of the previous three items in one location, adequate memory is allocated in SDRAM, and they are subsequently stored in SDRAM. The code used to pack the entire sensor data together in this manner is shown in Figure 5.3. Also include in the code to pack sensor data is a variable that tells the size of the packed sensor data, a necessary component for writing to flash memory.

With the sensor data successfully packed, *RemoteFlashSensing* initiates the writing phase by first erasing a predefined flash address. Following erasing of the address, the sensor data can be written into flash. Both the reading and writing of flash are handled by commands provided in the "FlashC" interface. The flash writing process relies on knowledge of the size of the sensor data, which is why the size was also stored in the packing phase. The code to erase the address and write the data to flash is shown in Figure 5.4. At this point in the operation, the sensor data is safely stored in flash memory, awaiting its retrieval from a local sensor node.

```
command result_t FlashService.packSensorData(SensorData *sdata2, uint8_t
**data, uint32_t *size)
{
	uint8_t i;
	uint32_t sdidx, len;
	uint8_t *sdata1;
	SensorData *sdata;

	for (i = 0, sdidx = 0; i < MAX_SENSOR_CHANNELS; i++) {
		sdidx += sdata2->channels[i].size * sizeof (uint16_t);
	}
	if (!syncsens) {
		sdidx += sdata2->timestamps.size * sizeof (uint32_t);
	}
	len = sizeof (uint32_t) + sizeof (SensorData) + sdidx;
	if ((sdata1 = (uint8_t *)sdmalloc(len)) == NULL) {
		trace(DBG_USR1, "ERROR: sdmalloc failed\r\n");
		call Leds.set(7);
		return FAIL;
	}
	trace(DBG_USR1, "sdmalloc complete\r\n");
	*(uint32_t *)sdata1 = len;
	sdata = (SensorData *) (sdata1 + sizeof (uint32_t));
	memcpy(sdata, sdata2, sizeof (SensorData));
	trace(DBG_USR1, "memcpy complete\r\n");
	sdidx = sizeof (uint32_t) + sizeof (SensorData);
	for (i = 0; i < MAX_SENSOR_CHANNELS; i++) {
		if (sdata2->channels[i].size == 0) {
			continue;
		}
		len = sdata2->channels[i].size * sizeof (int16_t);
		memcpy(sdata1 + sdidx, sdata2->channels[i].data,len);
		sdata->channels[i].data = (int16_t *)(sdata + sdidx);
		sdidx += len;
	}
	sdata->timestamps.data = NULL;
	if (sdata2->timestamps.size > 0) {
		len = sdata2->timestamps.size * sizeof (uint32_t);
		memcpy(sdata1 + sdidx, sdata2->timestamps.data, len);
		sdidx += len;
	}
	*data = sdata1;
	*size = sdidx + sizeof (uint32_t);
	*(uint32_t *)sdata1 = *size;
	return SUCCESS;
}
```

Figure 5.3 code for packing sensor data in *RemoteFlashSensing*.

```
command result_t FlashService.writeFlashData(SensorData *sdata, uint8_t *indx)
{
	uint8_t i, j;
	uint32_t size;
	uint8_t *sdata1;

	*indx = 0;
	call FlashService.packSensorData(sdata, &sdata1, &size);

	trace(DBG_USR1, "Erasing Flash...\r\n");
	if (call Flash.erase((uint32_t)(FlashAdrArray[1])) != SUCCESS) {
		trace(DBG_USR1, "ERROR: Flash erase failed\r\n");
		return FAIL;
	}

	trace(DBG_USR1, "Writing to Flash...\r\n");
	if (call Flash.write((uint32_t)(FlashAdrArray[1]), (uint8_t *)sdata1,
			size) != SUCCESS) {
		trace(DBG_USR1, "ERROR: Flash write failed\r\n");
		return FAIL;
	}

	*indx = 1;

	// Finished writing to flash
	trace(DBG_USR1, "Flash write complete.\r\n");
}
```

Figure 5.4 code for writing data into flash memory in *RemoteFlashSensing.*

When the local node is ready to retrieve the stored sensor data, a command is given to initiate the reading phase of the operation. This command is part of the "RemoteCommand" utility, and is called "RemoteFlashData". Full details of "RemoteCommand" are given in Sim and Spencer (2009). Upon initiation of the reading phase, the local node commands the remote node to read the data from a predefined flash address as shown in Figure 5.5. After reading from flash, the remote node unpacks the data in a manner opposite to that in which it was packed, as shown in the code in Figure 5.6. This unpacked data is stored in SDRAM on the remote node until it can be packed again and sent to the local node.

Once the data is read from flash and unpacked on the remote node, the command, "RemoteFlashData," utilizes the previously mentioned packing command to pack the data and send it to the local node, where it is unpacked and the measured data is saved to an output file. With the background outlined in this section, the next section details how to operate the *RemoteFlashData* application.

```
command result_t FlashService.readFlashData(SensorData **sdata, uint8_t indx)
{
	uint8_t i, j;
	uint8_t *fdata;

	fdata = (uint8_t *)(FlashAdrArray[1]);
	trace(DBG_USR1, "read flash data\r\n");

	call FlashService.unpackSensorData(sdata, fdata);

	return SUCCESS;

}
```

Figure 5.5 code to read from flash in *RemoteFlashSensing.*

```
command result_t FlashService.unpackSensorData(SensorData **sdata1, uint8_t
*data1)
{
       uint8_t i, *data, *sdata;
       uint32_t sdidx, len;
       SensorData *sd1;

       sd1 = (SensorData *)sdmalloc(sizeof (SensorData));
       *sdata1 = sd1;
       data = data1 + sizeof(uint32_t);
       memcpy(sd1, data, sizeof (SensorData));
       len = *(uint32_t *)data1;
       trace(DBG_USR1, "len = %u \r\n", len);
        if ((sdata = sdmalloc(len - sizeof (SensorData) - sizeof (uint32_t)))
== NULL) {
            trace(DBG_USR1, "ERROR: sdmalloc failed\r\n");
            return FAIL;
        }
        memcpy(sdata, data + sizeof (SensorData), len - sizeof (SensorData) -
sizeof (uint32_t));
        sdidx = 0;
        for (i = 0; i < MAX_SENSOR_CHANNELS; i++) {
            if (sd1->channels[i].size == 0) {
                continue;
            }
            sd1->channels[i].data = (int16_t *)(sdata + sdidx);
            sdidx += sd1->channels[i].size * sizeof (int16_t);
        }
        if (syncsens) {
            sd1->timestamps.data = NULL;
        }
        else {
            sd1->timestamps.data = (uint32_t *)(sdata + sdidx);
        }
             return SUCCESS;
}
```

Figure 5.6 code to unpack sensor data in *RemoteFlashSensing.*

5.2.2 How to operate *RemoteFlashSensing*

Step 1: Preparation

The first step in operating the *RemoteFlashSensing* application is to compile the source code to a binary file, and install it on each Imote2 sensor. To do this, one should open a cygwin window and change directories to the location of *RemoteFlashSensing*:

```
cd $SHMROOT/tools/RemoteFlashSensing
```

Using a USB-MiniB cable to connect the Imote2 to the computer, run the following command to compile and install the application on one node:

```
make imote2 install
```

The previous command will compile *RemoteFlashSensing*, and subsequently install the binary file on the Imote2. For the remaining Imote2 sensor nodes, the following command should be executed, one node at a time, while connected to the computer:

```
make imote2 reinstall
```

The previous command skips the compiling operation, and directly installs the binary file on the sensor node.

With the installation of the application on each node complete, the next step is to configure the network by designating the gateway and leaf nodes. The gateway node can be any node with *RemoteFlashSensing* installed, and is the node used with the base station. The base station is prepared by connecting an Imote2 with a Crossbow IIB2400 interface board to a computer. Each leaf node should have both a sensor board and a battery board. To communicate between the base station and the gateway node, the following command is run in a cygwin window:

```
imote2comm –d COMy
```

The 'y' is `COMy` represents the highest port number listed in the Windows Device Manager. Pressing <Enter> a couple of times should reveal a BluSH prompt. If not, the port number may be incorrect. The process regarding this step is well described in the ISHMP User's guide found at http://shm.cs.uiuc.edu/Imote2forSHM.html. In order to write the measured data to an output file, the following command should be executed in a separate cygwin window:

```
imote2comm –o out.txt COMx
```

The 'x' is `COMx` represents the lowest port number listed in the Windows Device Manager.

Step 2: Running RemoteFlashSensing

To run *RemoteFlashSensing*, input the following at the BluSH prompt:

```
setNodes <nodeID> [nodeID] [nodeID]
```

where `nodeID` is the node ID of a leaf node given in decimal format. After receiving the message, "- node ids are set.", input the parameters for sensing:

```
getData channelMask numSamples samplingRate syncSensing
```

where '`channelMask`' specifies the sensing channels, '`numSamples`' is the number of data points to be measured per channel, '`samplingRate`' is the sampling rate in hertz, and

'syncSensing' is a flag to indicate whether or not synchronized data should be obtained (1 for synchronized sensing, and 0 for unsynchronized sensing). Note that there are a limited number of sampling rates from which to choose with each sensor board, as seen in Table 5.1. When the message, "- parameters are set." is seen in the BluSH, type:

```
startSensing
```

Network-wide time synchronization, if specified, will begin after the 'startSensing' input. The initial LED color, red, will turn to yellow. After synchronization, the network begins sensing, and the LEDs turn green. After sensing, the data is resampled, and the leaf nodes print the first 10 points for demonstration and for comparison with data acquired from flash. After printing the first ten points, all sensor data is stored in flash memory. Screenshots for *RemoteFlashSensing* are shown in Figure 5.7 (gateway node) and Figure 5.8 (leaf nodes).

Table 5.1 Sampling rates supported by sensor boards.

Crossbow ITS400CA		Crossbow ITS400CB		Illinois SHM-A board	
Sampling freq (Hz)	Cutoff freq (Hz)	Sampling freq (Hz)	Cutoff freq (Hz)	Sampling freq (Hz)	Cutoff freq (Hz)
280	70	40	10	10	4
560	140	160	40	50	20
1120	280	640	160	100	40
4480	1120	2560	640	280	70

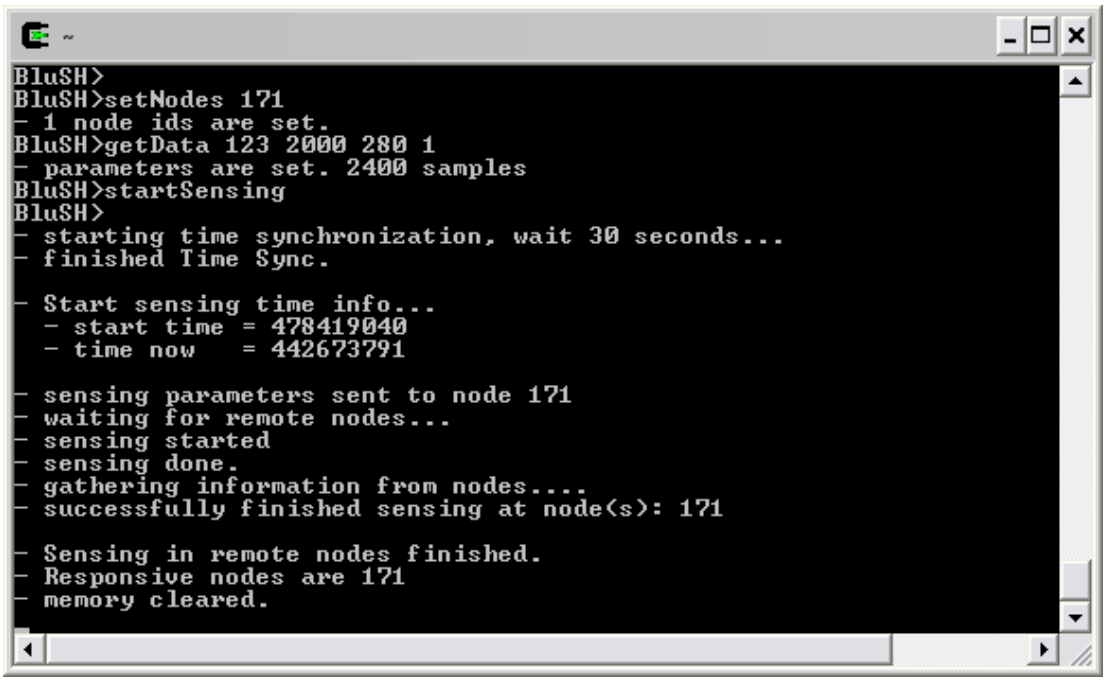


Figure 5.7 Output messages from the gateway node in *RemoteFlashSensing*.

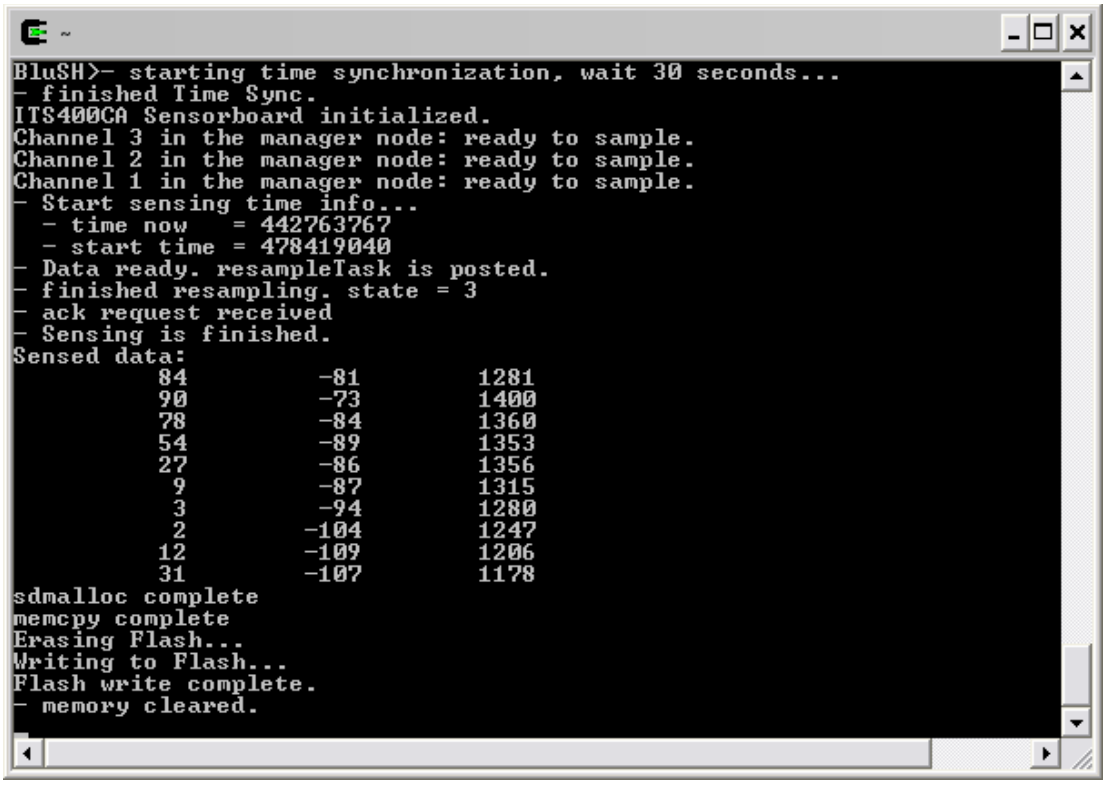


Figure 5.8 Output messages from the leaf nodes in *RemoteFlashSensing*.

Following the completion of flash storage on the leaf nodes, data can be retrieved at any time by the gateway node. To retrieve the data, the follow command is entered in the BluSH:

```
RemoteCommand FlashData <nodeID>
```

where `nodeID` is the node ID of the leaf node from which sensor data will be retrieved. The sensor data is sent back to the gateway node, at which point it is written into an output file. Screenshots of *RemoteCommand FlashData* are shown in Figure 5.9 (gateway node) and Figure 5.10 (leaf nodes). The first twenty lines of the output file written during the execution of *RemoteCommand FlashData* are shown in Figure 5.11. Note that the data from the leaf nodes prior to writing to flash (Figure 5.8) is exactly the same as the data retrieved from flash and sent to the gateway node (Figure 5.9 and Figure 5.11), showing that the sensor data remains uncorrupted during the entire execution of the *RemoteFlashSensing* application.

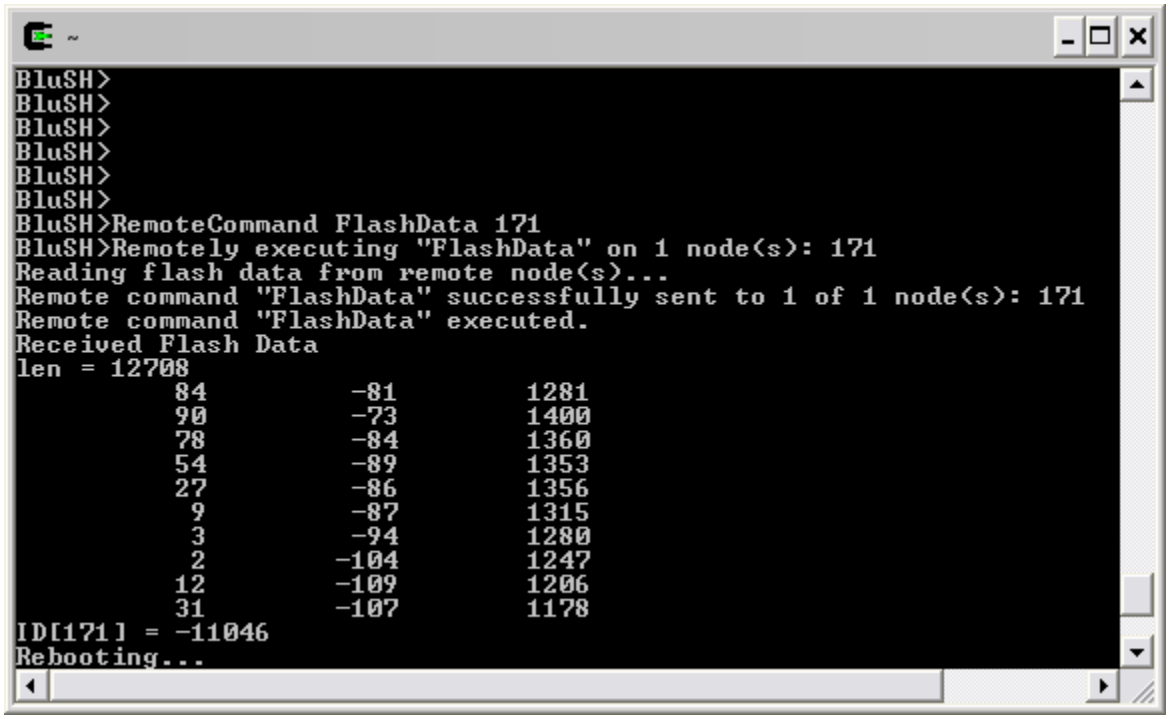


Figure 5.9 Output from gateway node executing *RemoteCommand FlashData*.

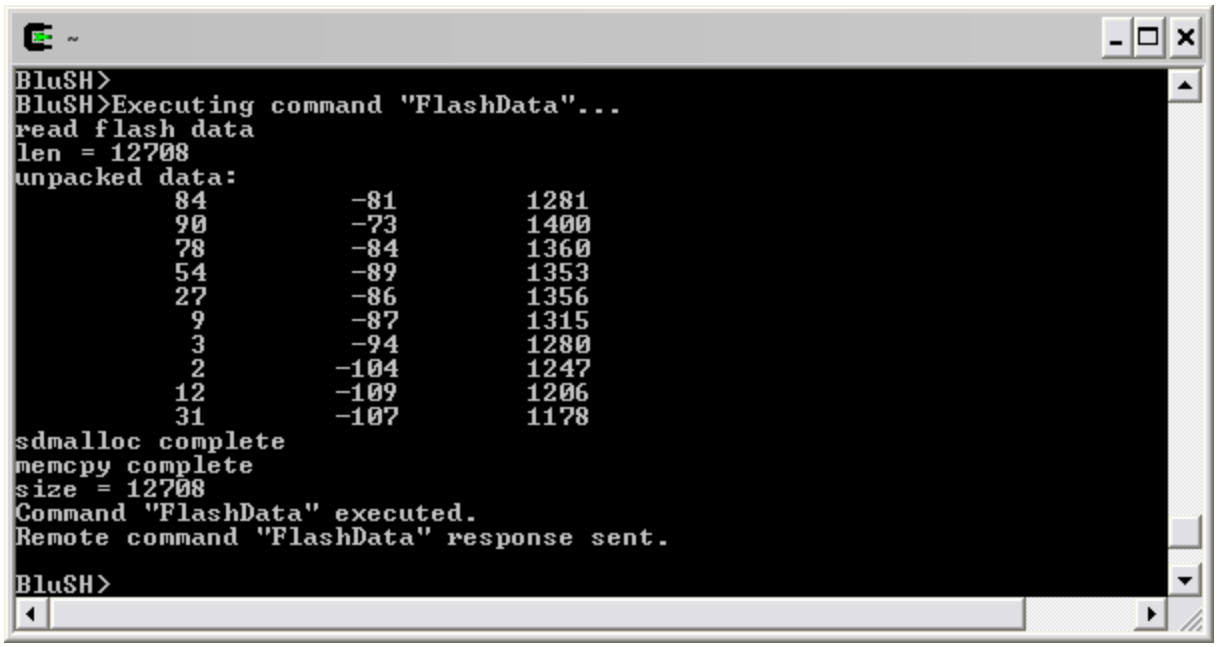


Figure 5.10 Output from leaf nodes executing *RemoteCommand FlashData.*

node	ch.1	ch.2	ch.3
171	84	-81	1281
171	90	-73	1400
171	78	-84	1360
171	54	-89	1353
171	27	-86	1356
171	9	-87	1315
171	3	-94	1280
171	2	-104	1247
171	12	-109	1206
171	31	-107	1178
171	44	-105	1156
171	46	-105	1133
171	48	-105	1111
171	48	-105	1084
171	43	-104	1052
171	34	-103	1023
171	27	-106	997
171	25	-112	974
171	21	-117	954
171	22	-124	938
171	44	-126	927
171	75	-116	919
171	87	-114	888
171	67	-128	829
171	30	-132	780

Figure 5.11 Output file (first 20 points) from *RemoteCommand FlashData.*

5.2.3 Power consumption of flash writing process

The *RemoteFlashSensing* application provides an effective means for storing sensor data on a leaf node for an extended period of time. Utilizing the flash writing process, however, can consume additional amounts of power, and this must be understood before incorporating *RemoteFlashSensing* into a WSSN.

To gain understanding of the additional power consumed by the flash writing process, laboratory tests were conducted to measure the current draw on a leaf node at each state of *RemoteFlashSensing*. The parameters for the test were as follows: 5000 samples on 3 channels sampled at 100Hz with time synchronization and resampling. The results (Figure 5.12) indicate that the current draw during flash writing is near 50mA, which is far less than the nearly 140mA experienced during the sensing and resampling stages. Additionally, the flash writing process takes only ~1 second to complete for all 15,000 data points. With this combination of low current consumption and short writing time, *RemoteFlashSensing* can be used in WSSNs without demanding excessive amounts of energy.

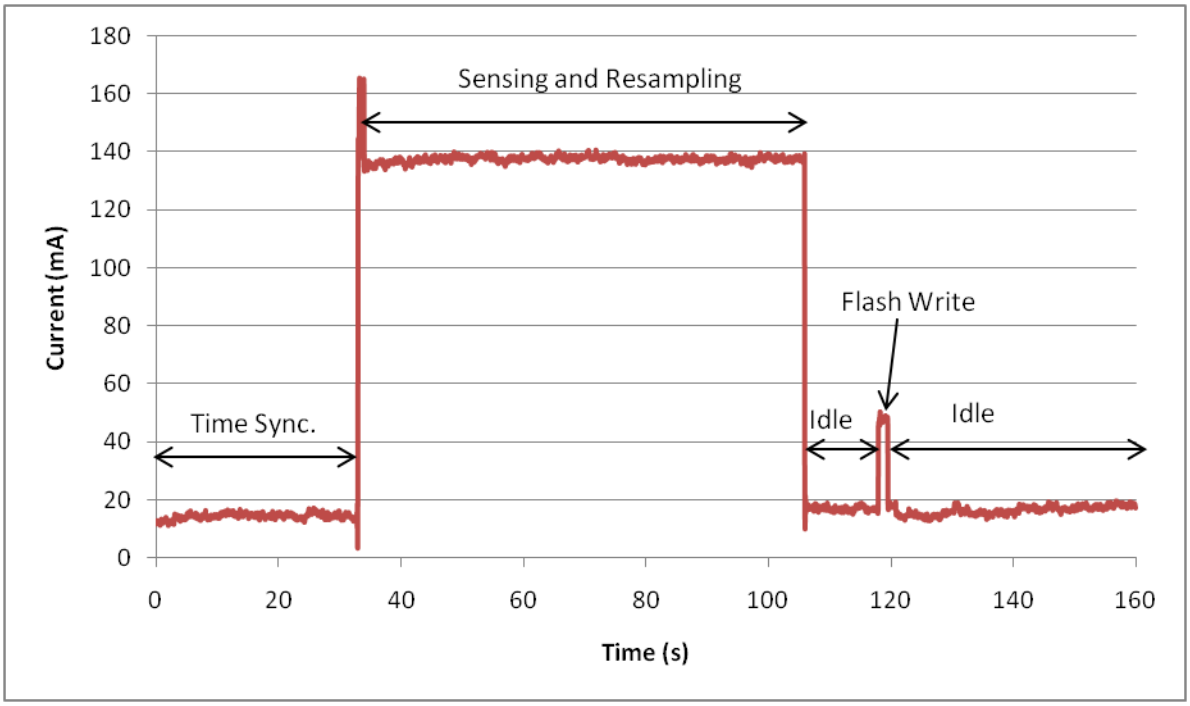


Figure 5.12 Current draw at each state during *RemoteFlashSensing*.

5.3 Summary

Utilizing non-volatile, flash memory for storing sensor data on leaf nodes represents one way to increase the autonomy of WSSNs; by utilizing flash storage, sensor data is preserved even in the event of a power failure or a communication failure. *RemoteFlashSensing* is an application that utilizes the sensing capabilities of *SensingUnit* and stores all of the sensor data into flash memory, without demanding significant amounts of power. The data remains uncorrupted, and is available for retrieval at any time thereafter. Using *RemoteFlashSensing* is an effective method for preserving sensor data even in the event of unforeseen failures.

Chapter 6 Full-scale test results

This chapter presents the results of a full-scale energy harvesting validation test which proves that the hardware modifications and the software developed by this research can be implemented to increase the efficacy of autonomous wireless smart sensor network (WSSN) operation. The full-scale test was conducted at the Jindo Bridge in South Korea, and reveals the suitability of the developed system for a cable-stayed bridge.

Also included in this chapter are the results of a full-scale validation of the *RemoteFlashSensing* application developed for this research, proving that it can be used to collect and preserve data on leaf nodes for an extended amount of time. The validation experiment was conducted at the Government Bridge in Rock Island, Illinois.

6.1 Jindo Bridge

The energy harvesting system developed in this research was tested on the Jindo Bridge in South Korea. The test was part of a deployment of 70 Imote2 sensor nodes with SHM-A sensor boards and Crossbow battery boards. Full details of the deployment are given in Rice and Spencer (2009), but details are also included in this report as a matter of completeness. The Jindo Bridge deployment is part of a collaborative effort between three institutions: the Korean Advanced Institute of Science and Technology, KAIST; the University of Tokyo in Japan; and the University of Illinois at Urbana-Champaign in the United States (Jang et al. 2009). The purpose is to show the suitability of the Imote2 platform, the SHM-A sensor board, the ISHMP software, and the solar energy harvesting system for full-scale structural health monitoring (SHM). Only the suitability of the solar energy harvesting system is discussed herein; the other items are discussed in Rice and Spencer (2009). The Jindo Bridge (Figure 6.2) is actually two separate cable-stayed bridges that connect Jindo Island to the far southwestern tip of the Korean Peninsula near Haenam (Figure 6.1). The main span of the bridge is approximately 344 meters in length. The original span was constructed in 1984, while the newer span was completed in 2005. The newer span is the focus of this deployment (Rice and Spencer 2009).

Figure 6.1 Location of the Jindo Bridge on the Korean Peninsula (left), between Jindo and Haenam (right) (Rice et al. 2009).

Figure 6.2 Twin spans of Jindo Bridge connecting Jindo Island with the Korean Peninsula with the newer span on the left (Rice et al. 2009).

6.1.1 Deployment goals

For the purpose of this research, the primary goal of the Jindo Bridge deployment is to assess the suitability of the solar energy harvesting system for use in an autonomous WSSN. The deployment is expected to reveal any shortcomings associated with the energy harvesting system, as well as to underscore the advantages realized when utilizing the solar energy harvester.

6.1.2 Deployment setup

In total, 70 Imote2 sensor nodes have been installed on the Jindo Bridge. Of those 70, however, only 8 are equipped with the components for solar energy harvesting. 5 of the solar harvesting nodes are on the cables, 2 are on top of pylons, and 1 is on the deck. For the initial deployment, the solar harvesting nodes are restricted to locations that would pose difficulty for maintenance and swapping batteries. In later phases, the desire is to deploy more solar energy harvesting nodes, pending adequate performance of this system. Following the initial sensor installation, the first phase of the deployment began on June 13, 2009. After making adjustments, a second preliminary phase (Phase II) began on June 23, 2009. The network was divided into two sub-networks, one on the Jindo side and one on the Haenam side. The Jindo side sub-network contains 37 nodes in total. 26 of these nodes are on the underside of the box-girder deck as shown in Figure 6.3, 3 nodes are on the pylon, and 8 nodes are on the cables. The Haenam side sub-network contains 33 nodes. 22 of these nodes are on the underside of the box-girder deck, 3 are on the pylon, and 8 are on the cables. The locations of the sensors are shown in Figure 6.5, including the location of all nodes with the energy harvesting feature. The sensor nodes equipped for solar energy harvesting are housed in PVC enclosures with the Li-polymer batteries and Antenova Titanis external antennas (Jang et al. 2009). This enclosure setup is shown in Figure 4.1 with the solar panel mounted on top of the enclosure.

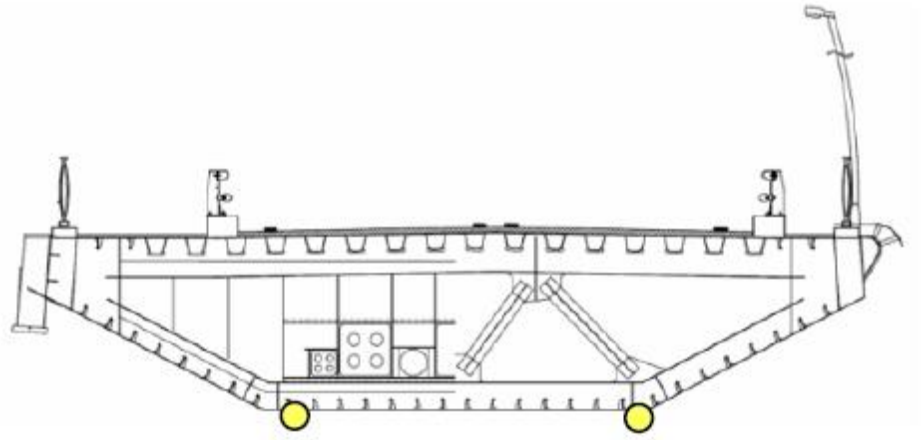


Figure 6.3 Sensor placement on either side of the box-girder deck.

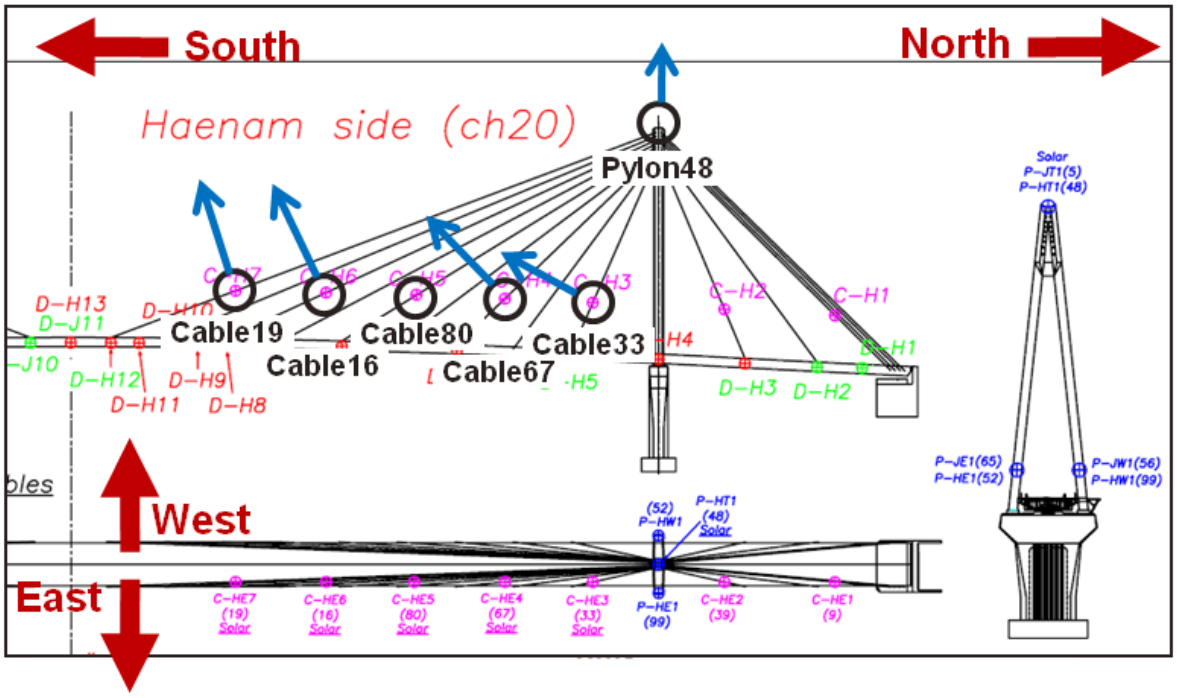


Figure 6.4 Solar sensor node locations on the Haenam side of the Jindo Bridge.

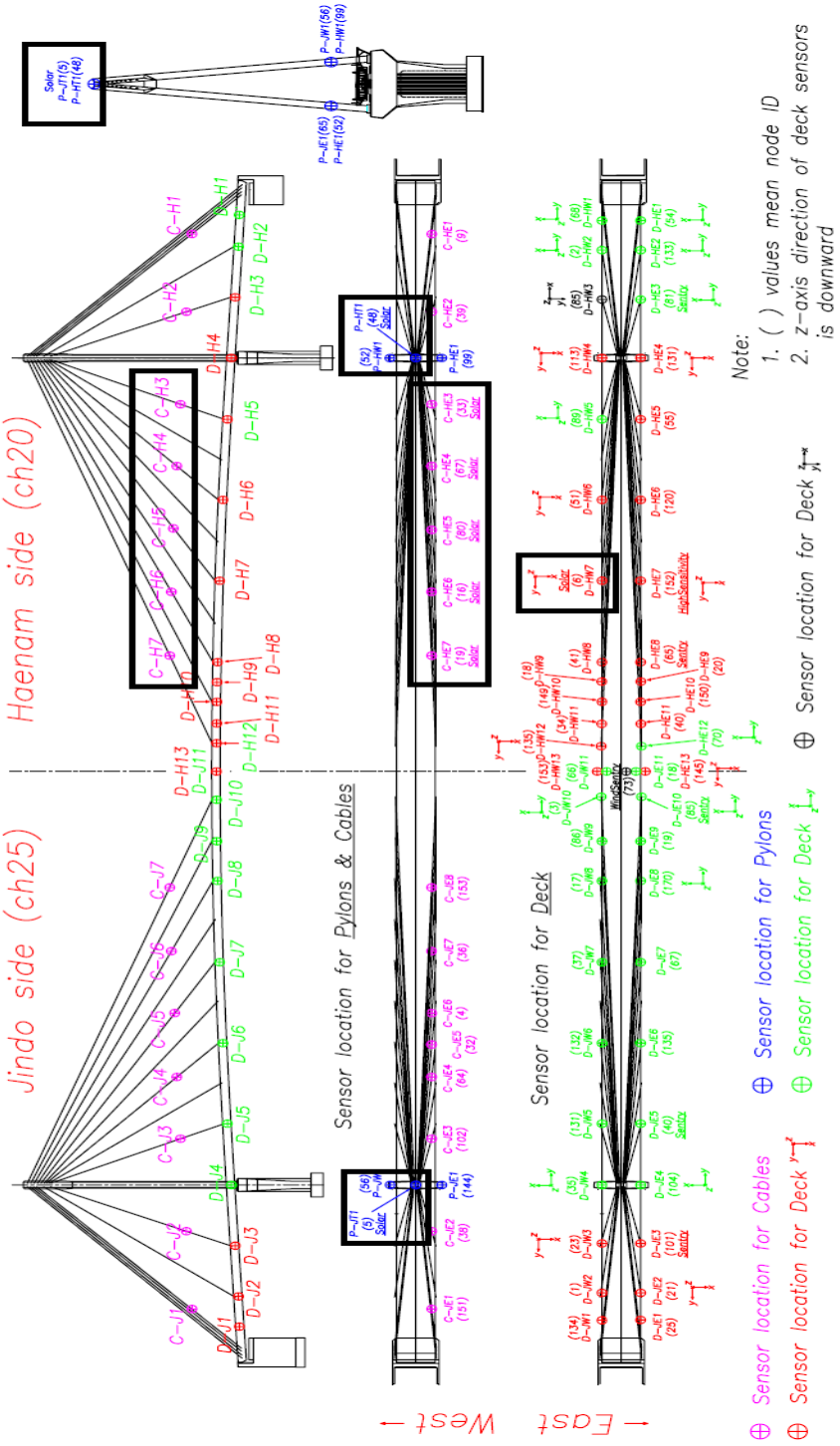


Figure 6.5 Sensor node locations on the Jindo Bridge on the Jindo side (left) and the Haenam side (right) (Jang et al. 2009).

Slightly more than one week after the initial phase, researchers returned to the Jindo Bridge to make adjustments to the system and commence the second phase. In this second phase (Phase II) of the deployment, the sensing parameters were adjusted to allow for longer data records, as compared with Phase I. The network parameters for Phase II of the deployment are shown in Table 6.1. The energy consumption resulting from these parameters is discussed in the next section.

Table 6.1 Jindo Bridge Phase II network parameters (Rice and Spencer 2009).

Category	Parameter	Value
Network Parameter	Network size	23 to 31
RemoteSensing parameters	Number of *RemoteSensing events per day*	1
	Sampling rate	50 Hz
	Channels sampled	3
	Number of data points	10,000
Network time intervals	Time synchronization wait time	30 sec
	Watchdog timer interval	60 min
	Broadcast message wait time	180 sec
SnoozeAlarm parameters	*SnoozeAlarm* wake/listen time	750 ms
	SnoozeAlarm sleep time	15 sec
	SnoozeAlarm duty cycle	4.76%
ThresholdSentry parameters	Sentry network size	2
	ThresholdSentry check interval	20 min
	ThresholdSentry sensing time	10 sec
	Threshold value	10 mg

6.1.3 Energy harvesting results

Following commencement of Phase II of the Jindo Bridge deployment, the battery voltages of the sensor nodes equipped with the solar energy harvesting system were recorded. These battery voltages, while not a perfect indicator of the capacity remaining in the battery, can give a good idea of the performance of the harvesting system. The battery voltages were recorded over an extended time period to assess the system's suitability for long-term SHM applications.

A chart of the recorded voltages is seen in Figure 6.6. The battery voltages were recorded from remote locations by utilizing the “Vbat” command available as one of the RemoteCommand options in the ISHMP Toolsuite.

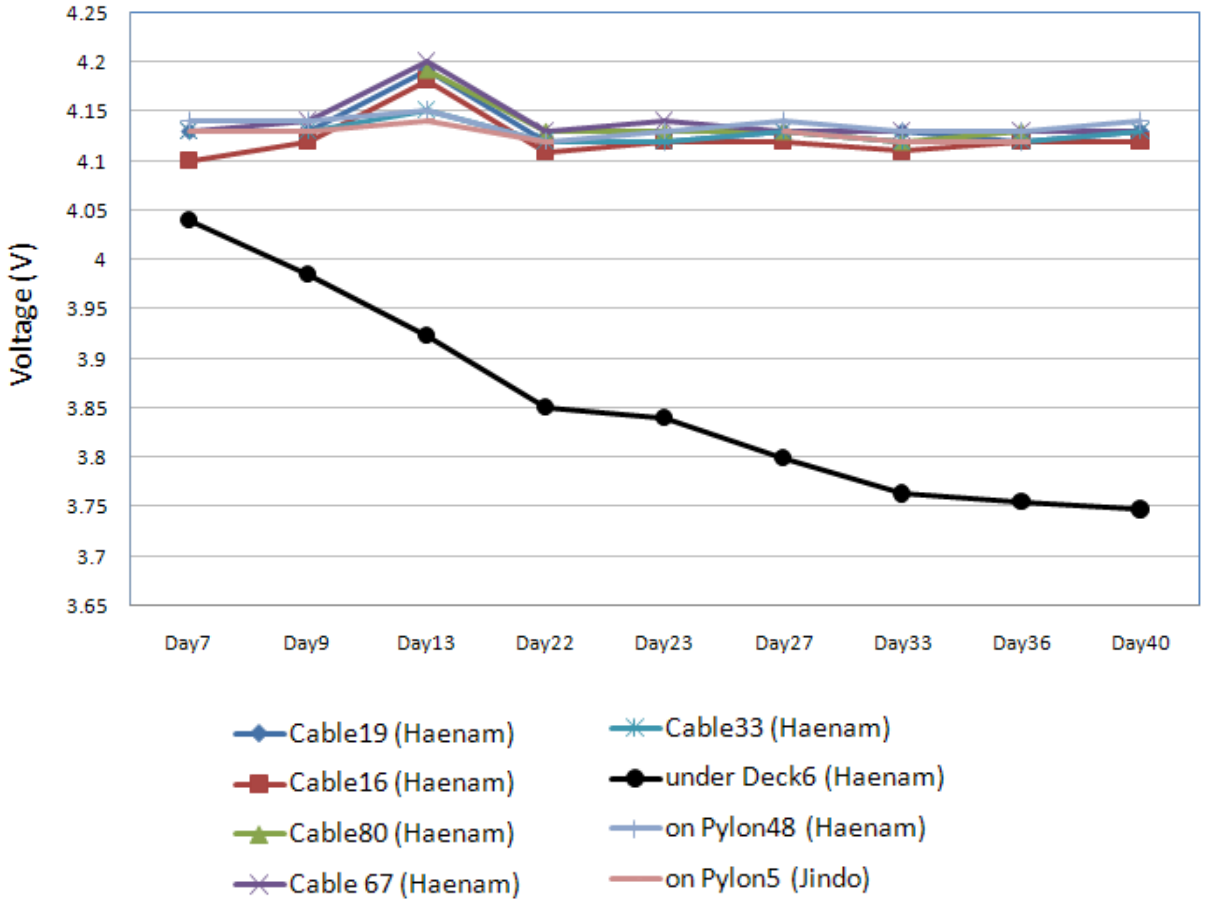
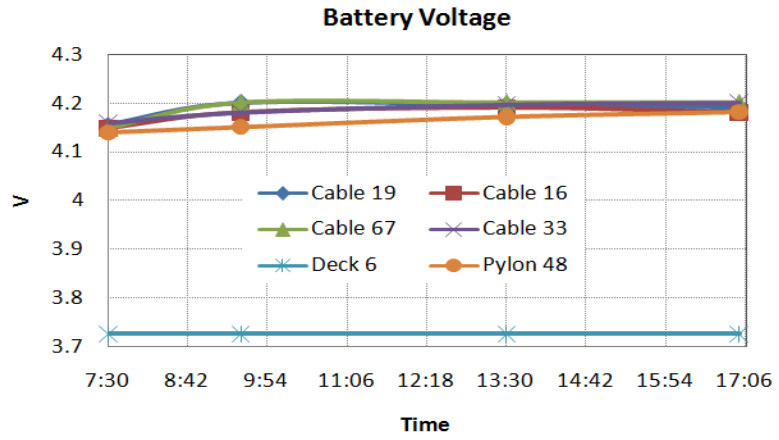


Figure 6.6 Battery voltages on solar energy nodes.

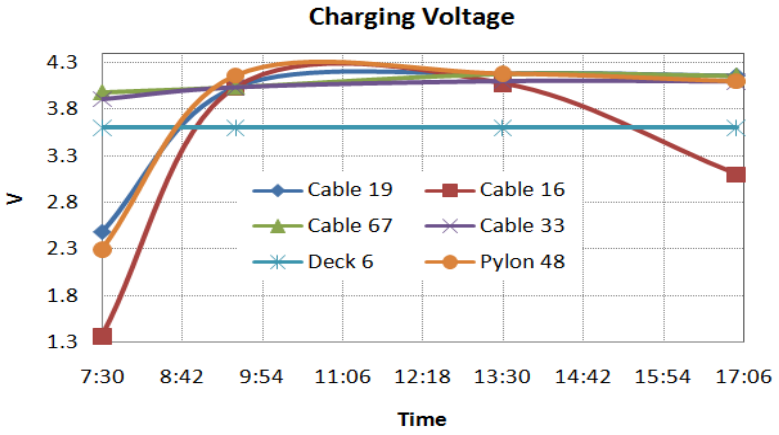
From Figure 6.6, two interesting observations can be made. First, seven of the eight solar energy sensor nodes are maintaining voltage levels above 4.1 volts, which suggests that for these nodes, the energy harvesting system is utilizing enough sunlight to keep them fully charged. Second, sensor node number 6 is exhibiting a continual decrease in voltage. Node number 6 is located under the deck on the Haenam side of the bridge, and it was speculated that the location inhibited sufficient amounts of light from reaching the solar panel. Judging by the placement of the deck nodes (Figure 6.3), this is likely the culprit, but a more detailed investigation of the problem was justified.

On October 15, 2009, researchers recorded the battery voltage, charging voltage, and charging current of six solar sensor nodes on the Haenam side of the Jindo Bridge to better understand the problem with deck sensor number 6. The measurements were taken remotely throughout the day using the “ChargeStatus” command, one of the available RemoteCommand options, and the results are shown in Figure 6.7. The locations of these six sensor nodes are shown in Figure 6.4 along with arrows indicating the direction in which the solar panel is facing.

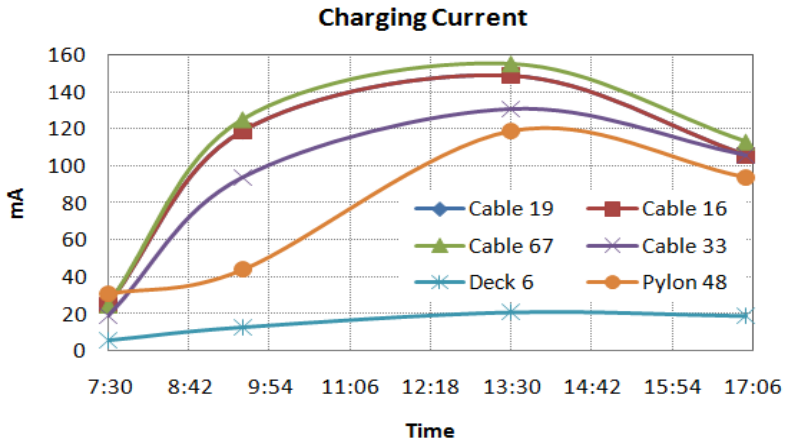
From the results in Figure 6.7, it is apparent that the deck node (node 6) is not positioned to take advantage of the current solar energy harvesting setup; despite the fact that October 15, 2009 was clear and sunny, the charging current yielded from the solar panel never exceeds 20mA. In addition to the low current production, the solar panel voltage output is merely 3.7 volts. With such low production values from the solar panel, the battery of node 6 never experiences a net charging status and thus continually loses capacity. Upon physical examination of the site, it is known that node 6 is mounted below the box-girder deck, and relies on reflected sunlight for most of its collected energy. On the other hand, the other five nodes on the Haenam side are positioned so that their solar panels receive direct sunlight at nearly all hours of the day, allowing their batteries to experience net charging when sunlight is adequate. From these observations, the conclusion is drawn that the current solar energy harvesting system is not adequate for charging batteries of the nodes at the deck level of the Jindo Bridge. The solar energy harvesters mounted on the cables and pylons perform more than sufficiently to provide perpetual power to the sensor nodes, but another solution may be needed to increase the lifetime of the nodes that do not receive direct sunlight, such as those mounted at the deck level under the box-girder.



(a)



(b)



(c)

Figure 6.7 Battery voltage (a), charging voltage (b), and charging current (c) of the solar sensor nodes on the Haenam side of the Jindo Bridge.

6.2 Government Bridge

The *RemoteFlashSensing* application developed for this research was validated at the Government Bridge in Rock Island, Illinois. Government Bridge (Figure 6.8), crossing the Mississippi River and connecting Rock Island, IL, and Davenport, IA (see Figure 6.9), is a historic monument to the late 19th Century. The bridge is maintained by the U.S. Army Corps of Engineers, and is owned by the United States Government. The structure has twin decks, one above the other. The top level accommodates railway traffic with two tracks and the lower deck is a roadway that accommodates regular vehicular traffic.

The draw span is about 365 feet in length and 60 feet in height at the center of the span. The pier of the draw span is placed between two canals that form part of Lock and Dam 15 on the Mississippi River. The draw span is supported at either end of the span when the bridge is in the fixed position, and both ends are cantilevered out from a turntable pivot section when the span is rotating. The bridge consists primarily of medium steel built-up members with several I-beams used for the bridge decks.

Figure 6.8 Government Bridge draw span in Rock Island, Illinois.

Figure 6.9 Location of swing span of the Government Bridge.

To validate the *RemoteFlashSensing* application on the Government Bridge, 4 leaf nodes were configured with the application and secured to the bridge on December 4th, 2009. The application was initiated using a gateway node, and the test was conducted to measure accelerations with 5000 data points on all three axes at a rate of 50 Hz with time synchronization and resampling of the data. Subsequent to completing the test, the nodes were powered down and taken back to the laboratory.

On December 7th, 2009 – three days after the nodes had been powered down – the acceleration data was requested from the leaf nodes, and it was successfully transmitted to the gateway node. The acceleration time history for two nodes in the transverse direction is shown in Figure 6.10. Note that the acceleration levels were too low to perform any modal analyses.

Without utilizing flash memory on the Imote2, the sensor data would be erased immediately when the node is powered down. With flash storage, however, the data can be preserved indefinitely. Successfully retrieving data stored in flash memory three days after the initial test proves that *RemoteFlashSensing* is an effective application for preserving sensor data for long periods of time.

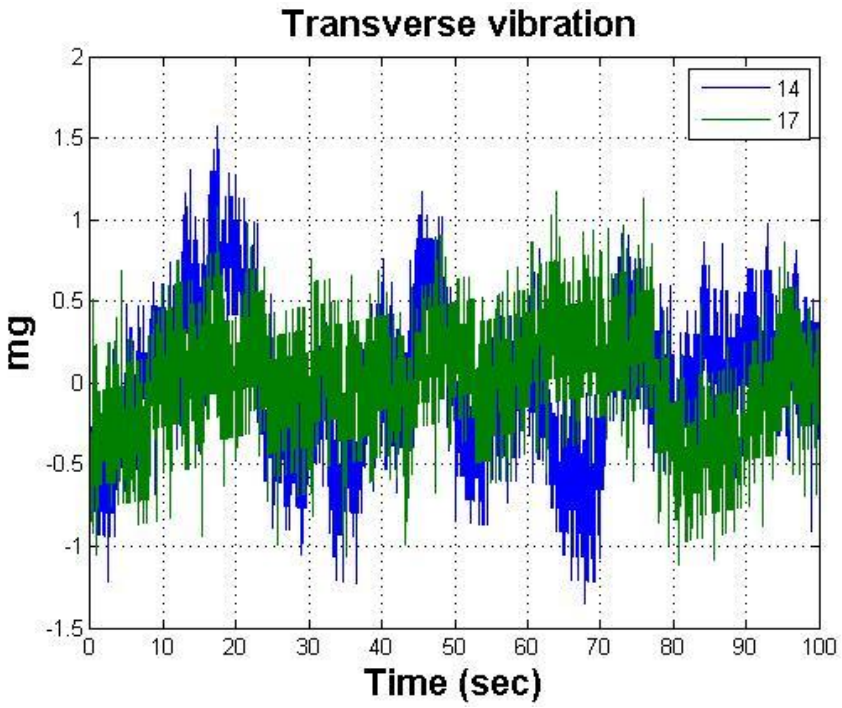


Figure 6.10 Transverse acceleration of Government Bridge.

6.3 Summary

The results presented in this chapter demonstrate how effective the solar energy harvesting system is at increasing the autonomy of WSSNs. The solar panel selection, the hardware modifications, and the software modifications all culminated to create a successful energy harvesting system that was used on the first autonomous, large-scale deployment of a WSSN for structural monitoring.

The energy harvesting system proved successful for well-exposed sensor nodes, but was unable to create enough energy to supply the shaded node with ample amounts of power. A new approach to the energy harvesting system is needed for shaded or partially shaded nodes, but the results of the Jindo Bridge test provide a basis for future developments in this area.

Furthermore, the results from the Government Bridge short-term deployment demonstrate that *RemoteFlashSensing* is an application that can be used to preserve sensor data and store it on leaf nodes for extended periods of time, making it ideal for use in SHM applications.

Chapter 7 Conclusion and future studies

7.1 Conclusion

The research detailed in this report has increased the autonomy of wireless smart sensor networks (WSSNs) used for structural health monitoring (SHM) purposes. The energy harvesting system developed for this research has been experimentally verified and can increase the lifetime of an entire network to reach that of its individual hardware components. Additionally, the software developed for storing and retrieving sensor data from flash memory can decrease the need for immediate transmission of data to a central location. The result of this research is a system that can operate as an autonomous entity for extended periods of time.

A wireless smart sensor (WSS) platform comparison has been given, ultimately revealing why the Imote2 is best suited for most high-demanding SHM applications. The high-performing processor, the high-capacity memory, and the scalability of the processor on the Imote2 make it the top choice for SHM applications. Its superior performance, however, causes the Imote2 to demand more energy than other platforms, suggesting a need for a smart power management strategy. Background on the efforts to reduce the energy consumption of the Imote2 has been given, and these efforts make up one portion of a smart power management strategy.

The need to approach the power management problem from the power supply side has been stated. Comprehensive background on the previous efforts to harvest ambient energy for use in an SHM network has been given, revealing the potential of energy harvesting. Despite the previous energy harvesting efforts, none have been shown to be adequate for the Imote2. A critical need to develop an energy harvesting system for the Imote2 was thus identified.

Beyond energy harvesting, the traditional scheme for collecting sensor data in networks of WSSs has been outlined. The traditional scheme reveals potential problems with the preservation of data if the network's nodes experience power problems or communication issues, identifying a need to allow network nodes to store sensor data on non-volatile memory so that it is preserved long-term until a gateway node calls for its retrieval.

A detailed description of solar energy harvesting using photovoltaic panels has been presented in chapter 3. The characteristics of solar panels and various rechargeable batteries

have been detailed. The implementation of a solar energy harvesting system has been discussed in chapter 4. The solar panel was selected based on its output characteristics and the energy demands of the Imote2. The rechargeable battery, a Lithium-polymer type, was selected because of its energy density, long lifetime, and low self-discharge rate.

The application, *RemoteFlashSensing*, has been introduced, further increasing the autonomy of WSSNs by eliminating the need to transmit sensor data back to a central computer for collection. This application can preserve sensor data in the event of power failure or communication loss, making it an option for redundancy in WSSNs. The code developed for this application, as wells as the charger control, is open-source so that other researchers and developers may modify it.

Finally, the solar energy harvesting system was experimentally validated on a test in South Korea. The Jindo Bridge deployment, still ongoing, is a test involving 70 sensor nodes, 8 of which are equipped with solar energy harvesters. The tests track the voltage of the rechargeable batteries over time, as well as the voltage and current from the solar panel. The test reveal that the energy harvesters are sufficient to supply perpetual power to the sensor nodes that are well exposed to the sun, but that the nodes covered by the box-girder deck may require a new approach to energy harvesting.

This research provided means to increase the autonomy of WSSNs for SHM applications. The solar energy harvester and the *RemoteFlashSensing* application both work toward better autonomy in separate ways. The result is a system with a long lifetime that can preserve sensor data in the network for long periods of time. The next section describes suggestions for further study to enhance the effectiveness of the energy harvester and software application, and further increase the autonomy of WSSNs.

7.2 Future studies

7.2.1 Energy harvesting enhancement

Although the energy harvester introduced in this report has proven viable for sensor nodes well exposed to the sunlight, the Jindo Bridge deployment shows that some sensor nodes will inevitably be sheltered from sunlight. To increase the autonomy of these nodes, and extend the overall network lifetime, a new approach to energy harvesting should be explored. Other

harvesting technologies, such as wind energy harvesting, may prove useful in areas with minimal sun exposure, but further research needs to be conducted to make them viable for the high-demand SHM applications on which the Imote2 is used.

7.2.2 Software enhancement

RemoteFlashSensing

The *RemoteFlashSensing* application is ideal for preserving sensor data in flash memory after a testing event. In its current form, however, it still must be initiated by a gateway node. One of the advantages of storing data in flash is that the leaf nodes are not required to transmit the sensor data back to a leaf node immediately, but requiring a test to be initiated at the gateway node slightly diminishes that advantage. To better improve the autonomy of the system, this application should be extended into something like *AutoMonitor* in the ISHMP Toolsuite. This extension would allow the *RemoteFlashSensing* to be initiated by other leaf nodes in the network as a result of a threshold event or a predetermined sensing time.

In addition to extending *RemoteFlashSensing* into something like *AutoMonitor*, the application should also be extended to allow multiple sensing events to be recorded in flash memory. The current form allows just one sensing event before data must be retrieved, but for better autonomy, the system should allow multiple sensing events. Perhaps multiple days' worth of sensor data can be collected before being retrieved by a gateway node. This extension would allow sensor nodes to sit at a test site for extended periods without any need for data collection or a base station, thus enhancing the autonomy of the WSSN.

7.2.3 Full-scale deployments

The full-scale deployment on the Jindo Bridge exhibits how informative such tests can be. By utilizing the solar energy harvester on the Jindo Bridge, the shortcomings and advantages of the system were well understood. More deployments in different conditions and on different types of structures should be sought in order to judge the efficacy of this solar energy harvesting system in various circumstances.

Additionally, the *RemoteFlashSensing* application should be installed on multiple sensors during another full-scale deployment for a longer period of time than that of the Government Bridge. This deployment will show the application's effectiveness and reliability in

a long-term, full-scale setting. Following such a full-scale deployment, further enhancements can be made to the application.

Chapter 8 References

Advanced Energy Group. (n.d.). *Solar Insolation for U.S. Major Cities*. From Solar4Power: http://www.solar4power.com/solar-power-insolation-window.html

Aldous, S. (2000, April 01). *How Solar Cells Work*. Retrieved from HowStuffWorks.com: http://science.howstuffworks.com/solar-cell.htm

Casciati, F., & Rossi, R. (2007). A power harvester for wireless sensing applications. *Struct. Control and Health Monit.; 14* , 649-659.

Çelebi, M. P. (2004). Seismic instrumentation of the Bill Emerson Memorial Mississippi River Bridge at Cape Girardeau (MO): A cooperative effort. *Proc. of 4th Int'l Seismic Highway Conf.,.* Memphis, TN.

Çelebi, M. (2002). *Seismic instrumentation of buildings (with emphasis on federal buildings).* United States Geological Survey (USGS).

Cho, S., Jo, H., Jang, S., Park, J., Jung, H.-J., Yun, C.-B., et al. (2009). *Structural Health Monitoring of a Cable-stayed Bridge Using Smart Sensor Technology: Data Analyses.*

Crossbow Technology, Inc. (2007). *Imote2 Hardware Reference Manual.*

Crossbow Technology, Inc. (2007). *MICA2 Wireless Measurement System.* San Jose, CA.

Dialog Semiconductor. (2005). *DA9030 Preliminary Product Data Sheet.*

Discenzo, F. M., Chung, D., & Loparo, K. A. (2006). Pump condition monitoring using self-powered wireless sensors. *Sound Vib., 40(5)* , 12-15.

Elvin, N., Elvin, A., & Choi, D. H. (2002). A self-powered damage detection sensor. *J. Strain Anal. Eng. Des., 38* , 115-124.

Glynne-Jones, P., & White, N. M. (2001). Self-powered systems: A review of energy sources. *Sensor Review, 21* , 91-97.

Ha, S., & Chang, F. K. (2005). Review of energy harvesting methodologies for potential SHM applications. *Proc., 2005 Int. Workshop on Structural Health Monitoring*, (pp. 1451-1460). Stanford, CA.

Inman, D. J., & Grisso, B. L. (2006). Towards autonomous sensing. *Proc. SPIE, 6174*, (pp. T1740-T1749).

James, E. P., Tudor, M. J., Beeby, S. P., Harris, N. R., Glynne-Jones, P., Ross, J. N., et al. (2004). An investigation of self-powered systems for condition monitoring applications. *Sensors & Actuators A, 110* , 171-176.

Jang, S. A., Jo, H., Cho, S., Mechitov, K., Rice, J. A., Sim, S.-H., et al. (2009). *Structural Health Monitoring of a Cable-stayed Bridge using Smart Sensor Technology: Deployment and Evaluation.*

Kling, R. (2003). Intel mote: An enhanced sensor network node. *Proc. of the International Workshop on Advanced Sensors, Structural Health Monitoring, and Smart Structures.*

Kling, R., Adler, R., Huang, J., Hummel, V., & Nachman, L. (2005). Intel mote-based sensor networks. *Structural Control and Health Monitoring, 12* , 469-479.

Lynch, J. P. (2006). A summary review of wireless sensors and sensor networks for structural health monitoring. *The Shock and Vibration Digest, 38(2)* , 91-128.

Mascarenas, D. L., Todd, M. D., Park, G., & Farrar, C. R. (2007). Development of an impedance-based wireless sensor node for structural health monitoring. *Smart Materials and Structures, 16* , 2137-2145.

Mateu, L., & Moll, F. (2005). Review of energy harvesting techniques and applications for microelectronics. *Proc. SPIE, Vol. 5837*, (pp. 359-373). Sevilla, Spain.

Paradiso, J. A., & Starner, T. (2005). Energy scavenging for mobile and wireless electronics. *Cognition, 4* , 18-27.

Park, G., Rosing, T., Todd, M. D., Farrar, C. R., & Hodgkiss, W. (2008). Energy Harvesting for Structural Health Monitoring Sensor Networks. *Journal of Infrastructure Systems, Vol. 14, No. 1* , 64-79.

Pfeifer, K. B., Leming, S. K., & Rumpf, A. N. (2001). *Embedded self-powered micro sensors for monitoring the surety of critical buildings and infrastructures.* Sandia National Laboratory, Albuquerque, N.M.: Sandia Rep. No. SAND2001-3619.

Polastre, J., Szewczyk, R., & Culler, D. Telos: Enabling Ultra-Low Power Wireless Research. *Proc. Fourth Int. Symposium on Information Processing in Sensor Networks.* Los Angeles, CA.

Qiwai, M. A., Thomas, J. P., Kellogg, J. C., & Baucon, J. (2004). Energy harvesting concepts for small electric unmanned systems. *Proc. SPIE, Vol. 5387*, (pp. 84-95).

Raghunathan, V., Kansal, A., Hsu, J., Friedman, J., & Srivastava, M. (2005). Design Considerations for Solar Energy Harvesting Wireless Embedded Systems. *IEEE International Conference on IPSN.*

Rice, J. A., & Billie F. Spencer, J. (2009). *Flexible Smart Sensor Framework for Autonomous Full-scale Structural Health Monitoring.* Urbana-Champaign: The Newmark Structural Engineering Laboratory.

Rice, J. A., Mechitov, K., Sim, S.-H., Nagayama, T., Jang, S., Kim, R., et al. (2009). *Flexible Smart Sensor Framework for Autonomous Structural Health Monitoring.*

Roundy, S., Wright, P. K., & Rabacy, J. M. (2004). *Energy Scavenging for Wireless Sensor Networks.* Norwell: Kluwer Academic Publishers.

Sim, S.-H., & Spencer, J. B. (2009). *Decentralized Strategies for Monitoring Structures using Wireless Smart Sensor Networks, NSEL-019.* Newmark Structural Engineering Laboratory.

Solarworld. (2009). *Solar panels*. From Solarworld: http://solarworld.com/SolarPanels.htm

Spencer Jr., B., Ruiz-Sandoval, M., & Kurata, N. (2004). Smart Sensing Technology: Opportunities and Challenges. *Structural Control and Health Monitoring 11* , 349-368.

Woodbank Communications Ltd. (2005). *Battery and Energy Technologies: Performance Characteristics*. From Electropaedia: http://www.mpoweruk.com/performance.htm